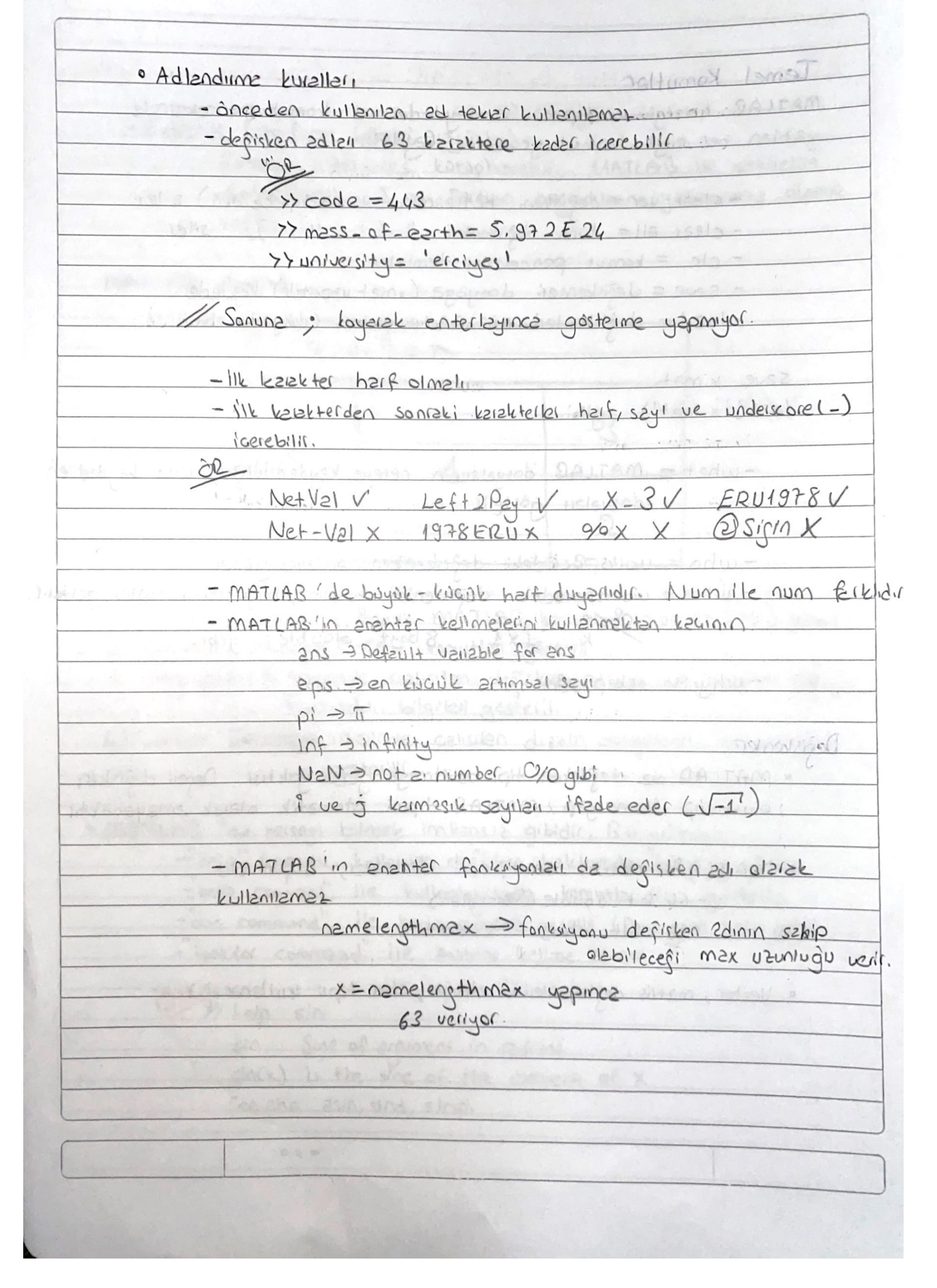


• Adlandırma kuralları

- önceden kullanılan ad tekrar kullanılamaz.
- değişken adları 63 karaktere kadar içerebilir

ÖR

```
>> code = 443
>> mass_of_earth = 5.972E24
>> university = 'erciyes'
```

// Sonuna ; koyarak enterlayınca gösterme yapmıyor.

- İlk karakter harf olmalı
- İlk karakterden sonraki karakterler harf, sayı ve underscore(_) içerebilir.

ÖR

NetVal ✓ Left2Pay ✓ X_3 ✓ ERU1978 ✓
Net-Val X 1978ERU X %X X @sigin X

- MATLAB'de büyük-küçük harf duyarlıdır. Num ile num farklıdır
- MATLAB'ın anahtar kelimelerini kullanmaktan kaçının.

ans → Default variable for ans
eps → en küçük artımsal sayı
pi → π
inf → infinity
NaN → not a number 0/0 gibi
i ve j karmaşık sayıları ifade eder ($\sqrt{-1}$)

- MATLAB'ın anahtar fonksiyonları da değişken adı olarak kullanılamaz

namelengthmax → fonksiyonu değişken adının sahip olabileceği max uzunluğu verir.

x = namelengthmax yapınca
63 veriyor.

Scalars

```
>> code=443
  code =
        443
```

```
>> c= 2.1 *21 -2*a
  c =
      42.1000
```

```
>> h = 22/7;
```

Çıktı Formatı

MATLAB sayıları floating-point değerler olarak saklar. ve onu öyle gösterir.
"format" ile gösterim değiştirilebilir.

```
>> format long
>> a = 1.123000123123123123;
>> a
 a =
     1.123000123123123  → 16 karakter (noktayla beraber)

>> a=100004545.99923423499111;
>> a
  a =
     1.000045459992342e+008   → 10^8

>> format short
>> a = 1.12300012312;
>> a
  a =
     1.1230  → 6 karakter

>> a=100004545.99923423499111;
>> a
  a =
     1.0000e+008
                → 10^8
```

Bunlar
- format long
- format short
- format short Eng
- format long Eng
- format hex   olabilir

Kullanılan formatı öğrenmek için
>> get(0, 'Format') yapılır.

// Sayıyı basamaklandırırken ondalık kısıma geçerken nokta(.) koy.

Vektörler

- MATLAB'de 2 tip array vardır
  - sayıların matrisi (double veya karmaşık)
  - cell array of objects (more advanced data structures)

- Row vector (satır vektörü); bracket ([ ]) arasındaki değerler virgülle veya boşlukla ayrılır.

```
>> row = [1 2 3 4];
>> row = [1, 2, 3, 4];
```

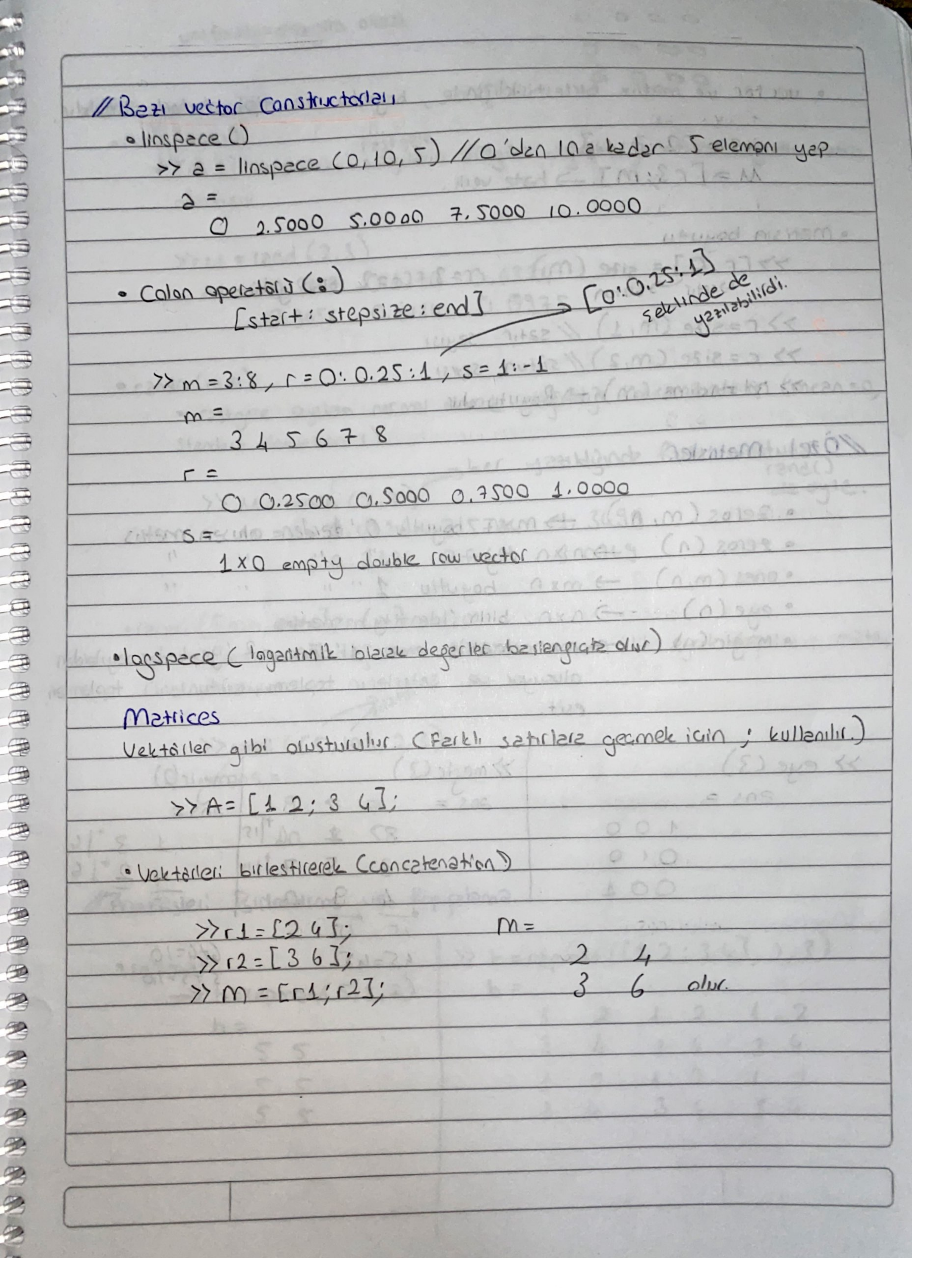
- Column vector (sütun vektörü); semicolon ile ayrılır.

```
>> col = [1; 2; 3; 4];
>> col = [1 2 3]';    // farklı çeşitte
```
(→ transpoze)

- Vektörün boyutu: length

```
>> length(col)
ans =
     3
```

* Boş vektör oluşturmak için;
  >> v = [ ]; yapılır.



// Bazı vector Constructorları

- linspace ()

```
>> a = linspace (0, 10, 5) // 0'dan 10'a kadar 5 elemanı yap
a =
   0   2.5000   5.0000   7.5000   10.0000
```

- Colon operatörü (:)

  [start : stepsize : end] → [0:0.25:1] şeklinde de yazılabilirdi.

```
>> m = 3:8 , r = 0: 0.25 : 1 , s = 1:-1
m =
   3 4 5 6 7 8
r =
   0 0.2500 0.5000 0.7500 1.0000
s =
   1 x 0 empty double row vector
```

- logspace (logaritmik olarak değerler başlangıçta olur)

Matrices

Vektörler gibi oluşturulur. (Farklı satırlara geçmek için ; kullanılır.)

```
>> A = [1 2; 3 4];
```

- Vektörleri birleştirerek (concatenation)

```
>> r1 = [2 4];
>> r2 = [3 6];
>> m = [r1; r2];
```

m =
2 4
3 6 olur.

• vector ve matrix birleştirildiğinde, boyutu ve tipi tutarlı olmalıdır.

r3 = [2 3 4]
M = [r3; m] → hata verir.

• Matrisin boyutu

>> [r, c] = size (m)   r=3 c=2 olur gibi
(r → satır, c → sütun)

>> r = size (m, 1) // satır sayısı
>> c = size (m, 2) // sütun sayısı
>> nd = ndims (m) → Boyutu dur.

// Özel matrisler

• zeros (m, n) → m x n boyutlu 0'lardan oluşan matris
• zeros (n) → n x n boyutlu " " " "
• ones (m, n) → m x n boyutlu 1 " " " "
• eye (n) → n x n birim (identity) matris
• magic (n) → n x n matris ama 1'den n^2'ye kadar sayılardan oluşuyor ve satırların toplamı, sütunların toplamları eşit.

>> eye (3)
ans =

1 0 0
0 1 0
0 0 1

>> magic (3)
ans =

8 1 6 | +15
3 5 7 | +15
4 9 2

+15 +15
45=45

magic(2)
ans =

1 3 | +4
4 2 | +6

5 5

6+4=10
5+5=10

uniformly=düzgün olarak

// Rastgele Matrisler

• rand (m,n)

• [0,1] aralığındaki rastgele sayıların düzgün dağıtıldığı m×n boyutlu matris oluşturur.

```
>>M = rand (2,3)
M= 0.8147  0.1270  0.6324
   0.9058  0.9134  0.0975
```

• randn (m,n)

• rastgele sayıların normal olarak dağıldığı m×n matris (mean=0, standart deviation 1)

```
>> M= randn (2,3)   → her yazıldığında farklı oluşturur rand() da öyle.
M = -0.4336  3.5784  -1.3499
     0.3426  2.7694   3.0349
```

• randi ([min, max], m,n)

– [min, max] değerleri arasından m×n'lik ayrık dağılmış matris oluşturur.

```
>>randi ([5,10], 2,3)   (dahil)
ans =
   9  5  7
  10  8  5
```

// Matrisleri Birleştirme ve Kopyalama

• repmat

```
>> b=repmat (5,3,2)
b =
   5 5
   5 5
   5 5
```

```
>> b=repmat ([1 2; 3 4], 2,3)
b =
   1 2 1 2 1 2
   3 4 3 4 3 4
   1 2 1 2 1 2
   3 4 3 4 3 4
```

• dikey birleştirme (vertcat'e bak)

```
>> v1 = [2 3 4];   v2 = [1 2 3];
>> x = [v1; v2];
   x =  2 3 4
        1 2 3
```

• yatay birleştirme (horzcat'e bak)

```
>> v1 = [2; 3; 4];   v2 = [1; 2; 3];
>> x = [v1 v2];
x =  2 1
     3 2
     4 3
```

• vertcat ile

```
>> v1 = [2, 3, 4];   v2 = [1 2 3];
>> x = vertcat(v1, v2)
   x =  2 3 4
        1 2 3
```

• horzcat ile

```
>> v1 = [2; 3; 4];   v2 = [1; 2; 3];
>> x = horzcat(v1, v2)
   x =  2 1
        3 2
        4 3
```

// Reshaping matrices

: operatörü

```
>> A = randi([1, 10], 4, 2)
A =
     5   7
    10   1
     8   9
    10  10
```

reshape

```
>> y = A(:)    → A'yı tek sıra halinde yazıyor.
y = 5
    10
    8
    10
    7
    1
    9
    10
```

```
>> B = reshape (y, 2, 4)
// y'yi 2x4 matrise dönüştürür

B =
    5   8   7   9
    10  10  1   10
```

```
>> M = reshape(linspace(11,18,8), [2,2,2])
// 11'den 18'e kadar 8 eleman oluştur ve bunu 2x2x2'lik matrise koy.

M(:,:,1) =  // 1. boyutu
        11  13
        12  14

M(:,:,2) = 15  17   olur.
           16  18
```

Arrayleri çevirmek (flip) için birkaç fonksiyon bulunmaktadır.

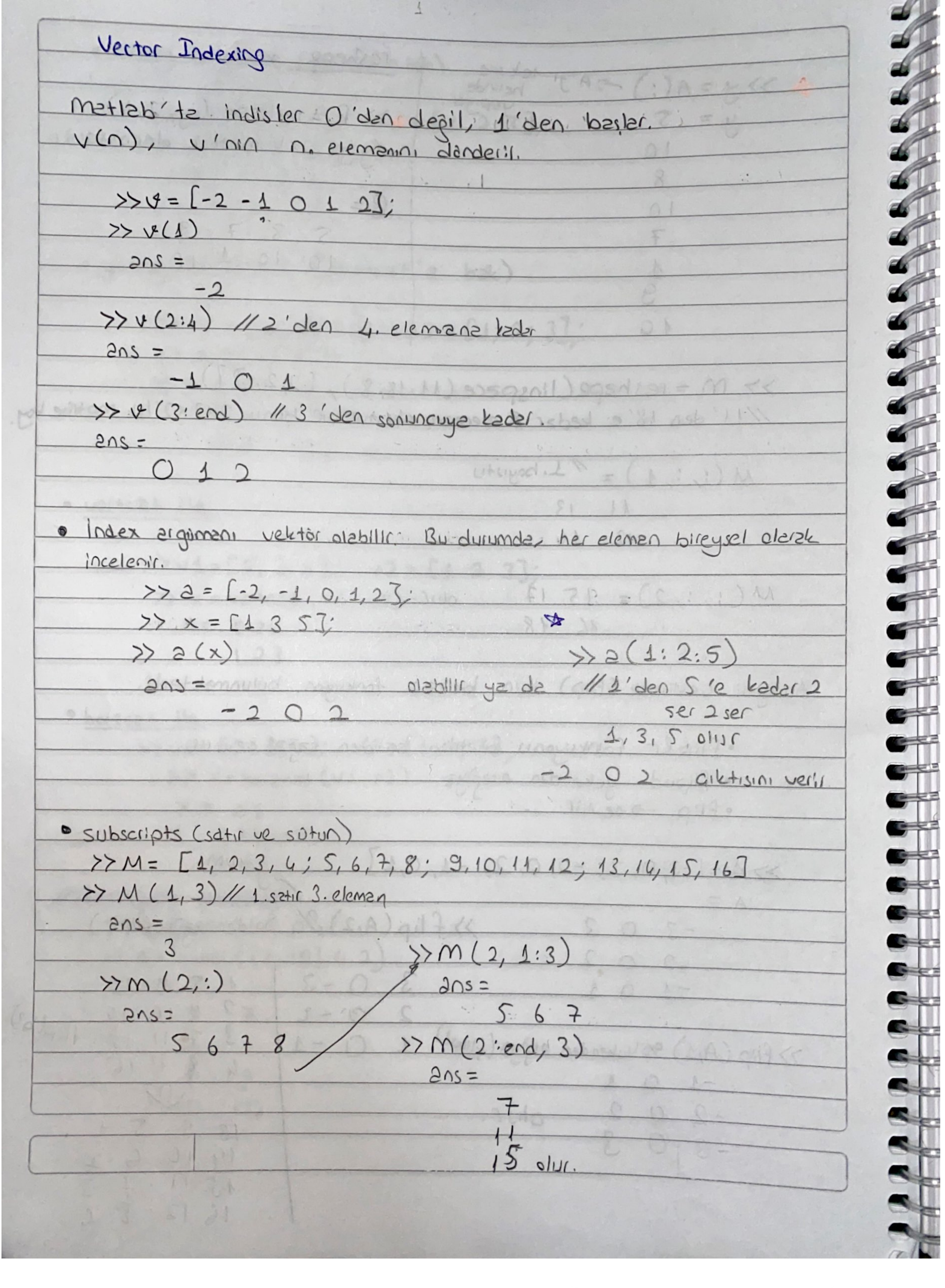
- fliplr fonksiyonu matrisi soldan sağa çevirir.
- flipud yukarıdan aşağıya
- flip → çevirir.

```
>> A = [-3, 0, 3; -2, 0, 2; -1, 0, 1]
A =
    -3  0  3
    -2  0  2
    -1  0  1
```

```
>> flip(A,2) % Soldan sağa (fliplr)     → yorum
    3  0  -3
    2  0  -2
    1  0  -1
```

```
>> flip (A,1) % Yukarıdan aşağıya (flipud)
    -1  0  1
    -2  0  2    olur.
    -3  0  3
```

```
 1   5   9  13
 2   6  10  14
 3   7  11  15     flip(A,2)
 4   8  12  16
        ⇓
13   9   5   1
14  10   6   2
15  11   7   3
16  12   8   4
```



Vector Indexing

Matlab'ta indisler 0'dan değil, 1'den başlar.
v(n), v'nin n. elemanını döndürür.

```
>> v = [-2 -1 0 1 2];
>> v(1)
  ans =
       -2
>> v(2:4)  // 2'den 4. elemana kadar
  ans =
      -1  0  1
>> v(3:end)  // 3'den sonuncuya kadar.
  ans =
      0  1  2
```

- index argümanı vektör olabilir. Bu durumda, her eleman bireysel olarak incelenir.

```
>> a = [-2, -1, 0, 1, 2];
>> x = [1 3 5];
>> a(x)
  ans =
      -2  0  2
```

olabilir ya da ☆

```
>> a(1:2:5)   // 1'den 5'e kadar 2 ser 2 ser
                 1, 3, 5, olur
     -2  0  2   çıktısını verir
```

- subscripts (satır ve sütun)

```
>> M = [1, 2, 3, 4; 5, 6, 7, 8; 9, 10, 11, 12; 13, 14, 15, 16]
>> M(1, 3)  // 1.satır 3. eleman
  ans =
      3
>> M(2,:)
  ans =
      5  6  7  8
>> M(2, 1:3)
  ans =
      5  6  7
>> M(2:end, 3)
  ans =
      7
      11
      15  olur.
```

- lineer indisler

```
>> M = [16 2 3 13; 5 11 10 8; 9 7 6 12; 4 14 15 1];
>> m(2)
  ans =
      5
>> m(7)
  ans =
      7
```

[1] 16	[5] 2	[9] 3	[13] 13
[2] 5	[6] 11	[10] 10	[14] 8
[3] 9	[7] 7	[11] 6	[15] 12
[4] 4	[8] 14	[12] 15	[16] 1

- Subscripts ve lineer indisler arasında convertler

```
>> ind = sub2ind(size(m), x, y) %   Subscript → lineer indis
>> [x,y] = ind2sub(size(m), ind) %   lineer ind → subscript
```

- Belli bir değerdeki yada rangedeki indisi bulma

```
>> L = [0, -1, 0 ; -1, 4, -1 ; 0, -1, 0];
>> ind = find(L<0)   // negatif elemanların indisini döndürür.
  ind =
      2
      4
      6
      8
```

>= <= kullanılabilir.

```
>> ind = find(L>0 & L<5)
  ind =
      5
```

İşlemler

- Aritmetik işlemler (+, -, *, /)

```
>> 7/45
>> (2+i) * 4/5  =>  ans = 1.6000 + 0.8000i
```

CamScanner ile tarandı

- Exponentiation (^) (üstel)

```
>> (3+2*j)^2
ans =
    24.000 + 10.0000i
```

- Karmaşıklığı önlemek için parantez kullan

```
>> ((2+3)*3)^0.5
```

- Multiplication is not implicit given parenthesis

```
>> 3(1+0.7)
     ↑
  Hata verir.
```

Transpoz

Transpoz işlemi sütunları satıra, satırları sütuna çevirir.

Ör

```
>> a=[-2:2]
a =
   -2   -1   0   1   2

>> a'
ans =
   -2
   -1
    0
    1
    2
```

```
>> a=[3,4; 1,2; 3,1]
a =
   3  4
   1  2
   3  1

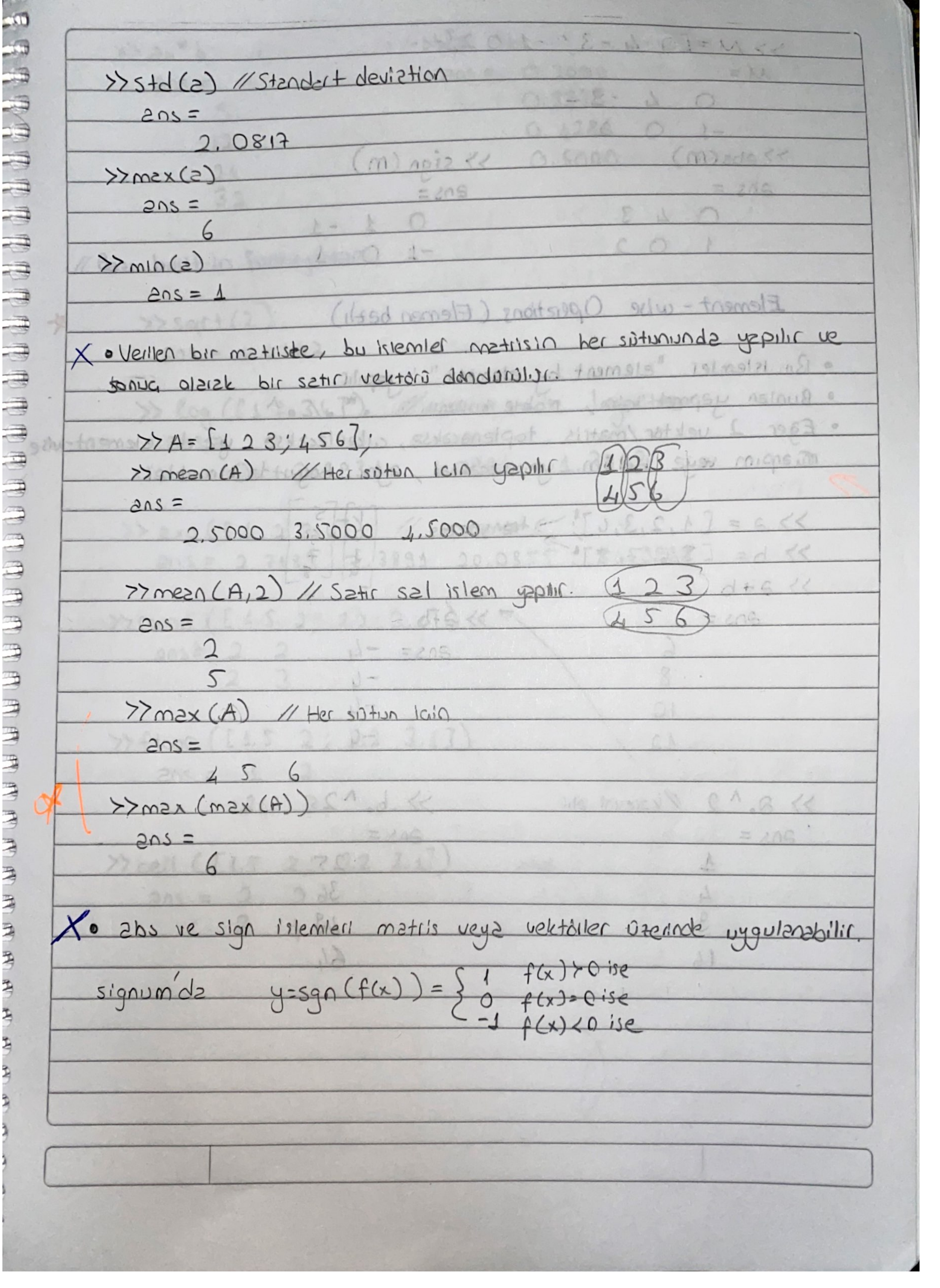
>> a'
ans =
   3  1  3
   4  2  1
```

Vektör işlemleri

Matlabın içinde vektörlerde işlem yapmamızı sağlayan fonksiyonlar vardır.

```
>> a=[1 4 6 3]
a =
   1  4  6  3
>> sum(a)
ans =
   14
```

```
>> mean(a)
ans =
   3.5000
>> var(a)  // varyans
ans =
   4.3333
```



```
>>std(a) //Standart deviation
   ans =
        2.0817
>>max(a)
   ans =
        6
>>min(a)
   ans = 1
```

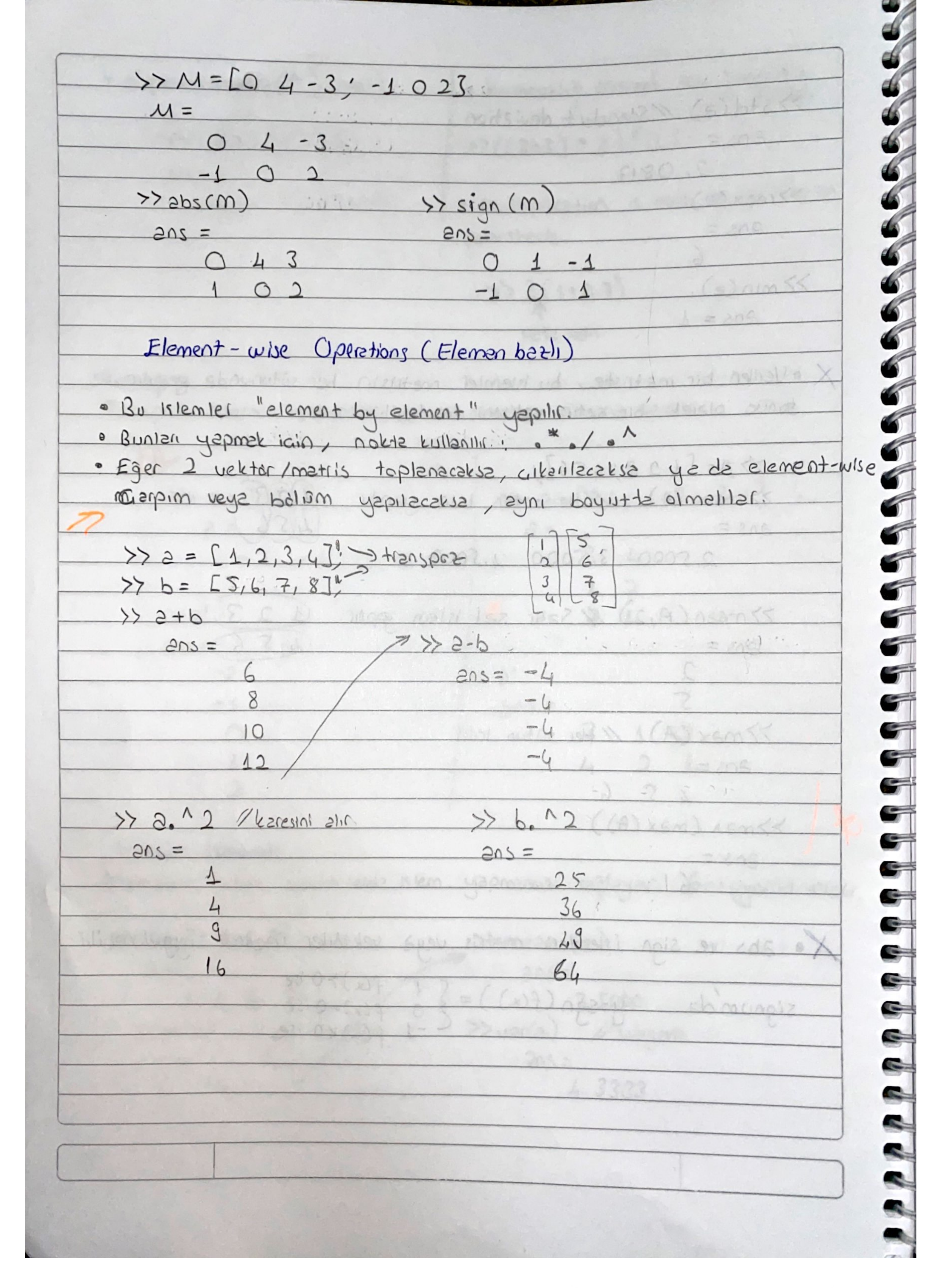
- Verilen bir matriste, bu işlemler matrisin her sütununda yapılır ve sonuç olarak bir satır vektörü döndürülür.

```
>>A=[1 2 3;4 5 6];
>>mean(A)      //Her sütun için yapılır
ans =
     2.5000   3.5000   4.5000

>>mean(A,2)  //Satırsal işlem yapılır.
ans =
     2
     5
>>max(A)    //Her sütun için
 ans =
     4   5   6
>>max(max(A))
 ans =
     6
```

- abs ve sign işlemleri matris veya vektörler üzerinde uygulanabilir.

signum'da
$$y=sgn(f(x))=\begin{cases}1 & f(x)>0 \text{ ise}\\ 0 & f(x)=0 \text{ ise}\\ -1 & f(x)<0 \text{ ise}\end{cases}$$



```
>> M = [0 4 -3; -1 0 2];
M =
     0   4  -3
    -1   0   2
```

```
>> abs(M)
ans =
     0   4   3
     1   0   2
```

```
>> sign(M)
ans =
     0   1  -1
    -1   0   1
```

Element-wise Operations (Eleman bazlı)

- Bu işlemler "element by element" yapılır.
- Bunları yapmak için, nokta kullanılır: .* ./ .^
- Eğer 2 vektör/matris toplanacaksa, çıkarılacaksa ya da element-wise çarpım veya bölüm yapılacaksa, aynı boyutta olmalılar.

```
>> a = [1,2,3,4]';   → transpoz
>> b = [5,6,7,8]';
```

$\begin{bmatrix}1\\2\\3\\4\end{bmatrix} \begin{bmatrix}5\\6\\7\\8\end{bmatrix}$

```
>> a+b
ans =
     6
     8
    10
    12
```

```
>> a-b
ans = -4
      -4
      -4
      -4
```

```
>> a.^2   // karesini alır
ans =
     1
     4
     9
    16
```

```
>> b.^2
ans =
    25
    36
    49
    64
```

CamScanner ile tarandı

```
>> a.*b
   ans =
          5
          12
          21
          32
```

```
>> a./b
   ans = 0.2000
         0.3333
         0.4286
         0.5000
```

// Bazı built-in fonksiyonlar

```
>> sqrt(2)
   ans = 1.4142

>> log([1 2 3 4])   // element-wise logaritma
   ans =
         0   0.6931   1.0986   1.3863

>> exp([1 2 3 4])   // exponential
   ans = 2.7183   7.3891   20.0855   54.5982

>> round([1.5 2; 2.2 3.1])
   ans = 2  2
         2  3

>> floor([1.5 2; 2.2 3.1])
   ans = 1  2
         2  3

>> ceil([1.5 2; 2.2 3.1])    olur
   ans = 2  2
         3  4
```

CamScanner ile tarandı

Mantıksal ve İlişkisel Operatörler

- Boolean değerler: 0 → false
  nonzero → true

- Bazı lojik operatörler;

<, <=, >, >=	küçük, büyük, küçük eşit --
== ~=	eşit, eşit değil
&	mantıksal AND
|	mantıksal OR
~	mantıksal NOT
all	all true
any	any true
xor	xor

Mantıksal indisleme

```
>> R = rand(5)
R =
    .  .  .  .  .
    .
    .
    .
    .
```
} 5x5 random matris oluşturur.

```
>> R(R<0.15)'   // R'nin 0.15'ten küçük elemanlarını alt indislerdeki
ans =              elemanları transpozu aldıktan sonra yazdır.
   0.1270  0.0975  0.1415  0.0357

>> isequal(R(R<0.15), R(find(R<0.15)))
ans =
   1
```

Find Metodu

- find nonzero değerlerin indisini dönderir. Yani bulduysa

- index = find (condition)

```
>> x = rand (1,10)
   x =
       0.4505 0.0838 0.2290 0.9133 0.1524 -->
   --> 0.8258 0.5383 0.9961 0.0782 0.4427

>> inds = find (x>0.4 & x<0.7)
   inds =
        1 7 10

>> x(inds)
   ans=
        0.4505   0.5383   0.4427
```
olur

BÖLÜM 2: Introduction to MATLAB: Scripts and Functions, Control Loops and Data Structures

M-Files

MATLAB programı içeren text dosyaları şu şekilde çağrılabilir;
- command window
- m-files

There are 2 kind of m-files
- scripts
- functions

Scriptlerde fonksiyonlarda saklandıkları komut dizileri tekrar kullanılmasına olanak sağlar.
- Scripts are simple: komutları aynen command line'a yazdığınız gibi saklar.

- Fonksiyonlar çok esnek ve daha kolay genişletilebilirdir.

1) Scripts

Scriptler;

- bir dizi halinde çalıştırılan komutlar koleksiyonudur.
- MATLAB editörde yazılmıştır.
- MATLAB dosyaları olarak kaydedilirler. (.m uzantılı)

Command window'dan MATLAB dosyası oluşturmak için New Script'e tıklanır.

Scriptleri açmak içinse command window'dan

```
>> open helloword.m
```

x.m → script adı

line numbers

% → yorum

Komutlar

şeklinde bulunur. (– → breakpoints)

NOT

Yorum;

% ile yorum yapılabilir.

ilk gelen yorum scriptin help dosyasıdır.

Uygun commentler zamandan tasarruf sağlar.

Scriptlerde bir input ve output yoksa, statik diyebiliriz.

Bir script dosyasında oluşturulan ve değiştirilen tüm variables, çalışmayı durdurduktan sonra bile çalışma alanında bulunur. (→ workspace)

verbose=ayrıntılı

Scriptler workspace deki bir değişkene erişip üzerinde oynamalar yapabilir.

Ör

coin.m adında bir script yazın. Bu scriptle yazı-tura oyununu tasarlayacaksınız.

Her bir atışta (toss) Heads veya Tails yazmalı.

```
% coin.m                          → help file
% a script that flips a coin and displays the output.

if rand < 0.5 % random sayı 1/2'den küçükse
        display('Heads');
else
        display('Tails');
end
```

NOT
- Burada scriptlerinizin başka kişiler veya kendiniz tarafından daha kolay anlaşılmasını sağlayacak bazı ipuçlarını göreceğiz.

- Script dosyaları header bölümünde şunları içermelidir.
  - program ne yapacak
  - yazarı kim
  - kimin tarafından yazıldı veya gözden geçirildi
  - Variable dictionary

- Programın anlaşılabilir olması için white spaceler kullan. (Kodlar arasını enterla)
- Comenentler kullan.
- Değişkenler için anlamlı ifadeler kullan. % image-size
- clear all ve clc komutlarını scriptin başında kullan.

2) Functions

- Fonksiyonlar scriptlere benzer, tek farkı içinde fonksiyon tanımlaması olmalıdır.

    function output = functionName(input)

- Fonksiyon içinde oluşturulan ancak döndürülmeyen (return edilmeyen) tüm değişkenler, işlevin çalışması durduktan sonra kaybolur.

ÖR

outputs / input

```
function [avg, sd, range] = stats(x)
%stats: avelagı, standart sapmayı ve verilen range'i hesaplar.

avg= mean(x);
sd = std(x);
range = [min(x); max(x)];

end
```

help

MATLAB'de fonksiyon yazmak için;

Cmd'ye save fonksiyon.m yap yada new → function file yap

(dosya adı ile aynı olmalı)

fonksiyon çağırma adı da aynı olmalı

bunu yaz ancak stats yerine fonksiyon yaz.

Cmd'den x = [3, 2] yapıp
fonksiyon (x) ile çağırılabilir.

User - Defined fonksiyonlar

must have the reserved word function ↑

function name should match m-file name. ↑

Inputs must be specified

function [out1, out2] = functionName(in1, in2)

birden fazla output bracket içinde belirtilmeli

- Return'a gerek yok. = MATLAB zaten fonksiyon tanımındaki yapılan değişken isimlerini döndürecektir.

- Variable scope = fonksiyon içinde oluşturulan ancak döndürülmeyen (return edilmeyen) tüm değişkenler, işlevin çalışması durduktan sonra kaybolur.

overloading

- MATLAB fonksiyonları genellikle overloaded tır.
  - Değişken sayıda input alabilir.
  - Değişken sayıda output döndürebilir.

```
>> a = zeros (2,4,8);   → 2x4 boyutunda 8 tane matris oluştur
>> D = size(a)          → 2, 4, 8 döndürür. elemanları 0 olsun
>> [m,n] = size(a)      → m=2  n=32 döndürür
>> [x,y,z] = size(a)    → x=2  y=4  z=8  döndürür
>> m2 = size(a, 2)      → 4 döndürür
            a, 1 → 2
            a, 3 → 8  döndürür.
```

• Kendi fonksiyonlarınızı bu şekilde değişken sayıda input ve output-larla overload edebilirsiniz.

Number of Inputs / Outputs

• nargin, çalıştırılan fonksiyondaki input değişkenlerinin sayısını döndürür.

```
function c = addme(a,b)
  switch nargin
        case 2
              c = a + b;
        case 1
              c = a + a;
        otherwise
              c = 0;
  end
end
```

Command Prompta

```
addme(3,4)
7
addme(3)
6
addme()
0        olur.
```

• nargout, çalıştırılan fonksiyondaki output değişkenlerinin sayısını döndürür.

Global Variables

- Fonksiyonlar, temel workspace'i kullanmazlar. Her fonksiyon kendine alt workspace'i vardır.

- Birkaç fonksiyon tek bir değişkeni paylaşmak gerekliliğindeyse, bu değişken global olarak tanımlanabilir.

- Herhangi bir fonksiyonda bu global değişkenin değiştirilmesi, diğer fonksiyonlara da yansıyacaktır.

```
global x
x = 5
```
CMD'de

```
function r = displayGlobalX
    global x
    val = x;
    disp(val);
end
```

buraya yollamasak da arada alınır.

Anonymous Functions

- Dosya oluşturulmadan fonksiyonlar
  - 0 veya fazla parametre alabilir.
  - İçice anonymous fonksiyonlara izin verilir.
  - Expression ve gerekli ifadeler gerekli

CMD

```
>> function = @(x,y) x^2 + y^2;
```
(function → fonksiyon adı olmalı.)

Daha sonra

```
>> fonksiyon adı (3,4)
```
= 25 olur.

Local Functions

- MATLAB dosyasında birden fazla fonksiyon bulunabilir.
- İlk fonksiyon main fonksiyondur.
  - Herhangi bir yerden çağırılabilir.

- Other functions local fonksiyonlardır.
  - Only callable from main function or other local functions in same file.
  - Enables modularity (large number of small functions) without creating a large number of files.
  - Unfavorable (elverişsiz) from code reusability point.

Control Loops

İlişkisel (Rational → relational) ve Logical Operatörs

★ Boolean değerlerde = zero = false
non zero = true

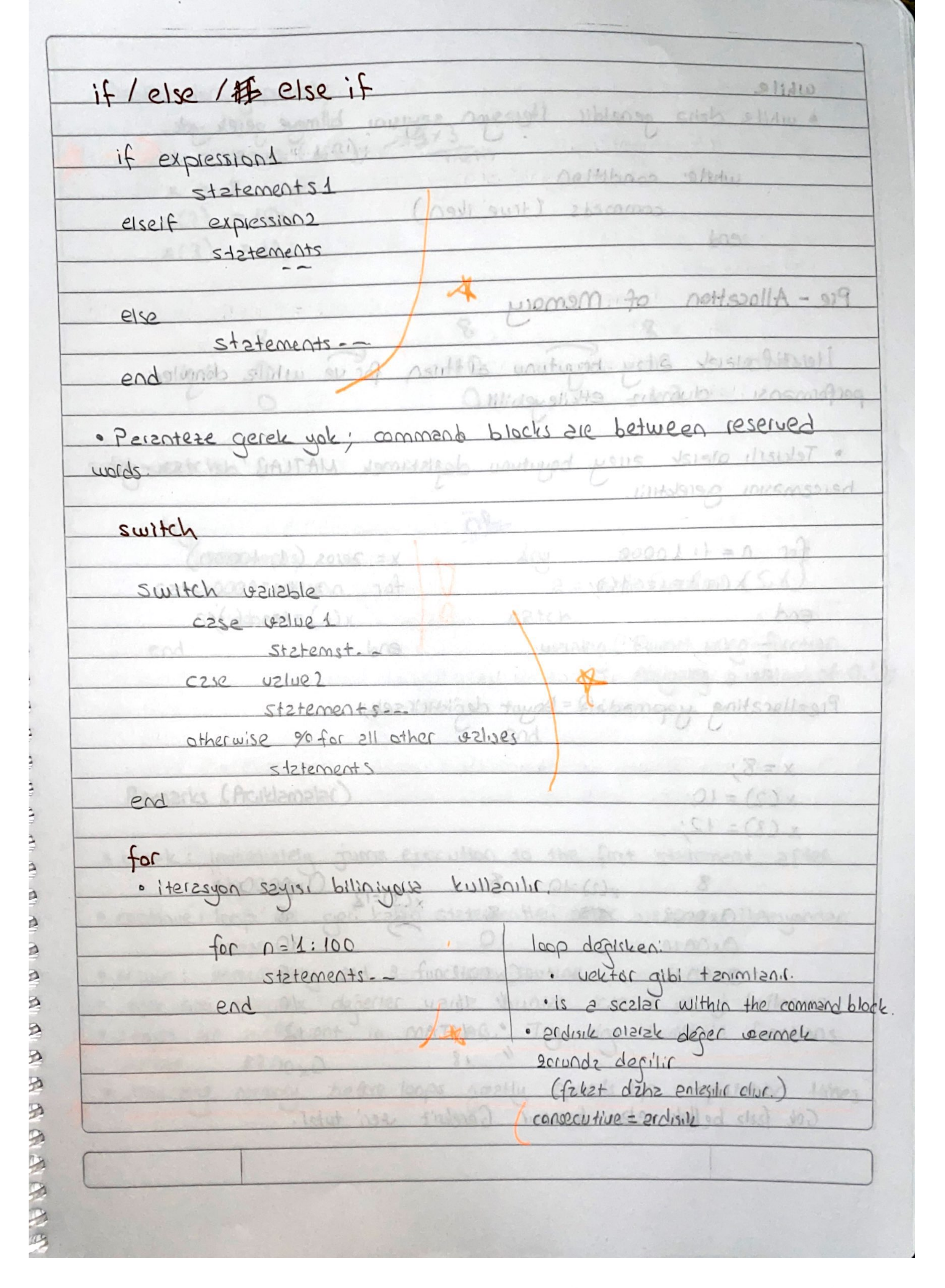
Bazı mantıksal operatörler

- <, <=, >, >= } less than, less than or equal to etc.
- ==, ~= } equal to, not equal to

Bazı ilişkisel Operatörler

- & → logical and
- | → logical or
- ~ → logical NOT
- all → all true
- any → any true
- xor → xor
- false
- find
- isLogical
- logical
- true

gibi



if / else / else if

```
if expression1
    statements1
elseif expression2
    statements
    --

else
    statements --
end
```

• Parantez gerek yok; command blocks are between reserved words.

switch

```
switch variable
    case value 1
        statemst. -
    case value2
        statements..
    otherwise  %for all other values
        statements
end
```

for

• iterasyon sayısı biliniyorsa kullanılır.

```
for n=1:100
    statements --
end
```

loop değişken:
• vektör gibi tanımlanır.
• is a scalar within the command block.
• ardışık olarak değer vermek zorunda değilir (fakat daha anlaşılır olur.)
(consecutive = ardışık

while

- while daha geneldir. İterasyon sayısını bilmeye gerek yok.

```
while condition
      commads (true iken)
end
```

Pre-Allocation of Memory

İteratif olarak array boyutunu arttıran for ve while döngüleri performansı olumsuz etkileyebilir.

- Tekrarlı olarak array boyutunu değiştirmek MATLAB'da extra time harcamasını gerektirir.

```
for n = 1:10000
    x(n) = rand();
end
```

```
x = zeros(1, 10000)
for n = 1:10000
    x(n) = rand();
end
```

Preallocating yapmadan boyut değiştirirsek;

```
x = 8;
x(2) = 10;
x(3) = 12;
```

8	x(2)=10	8	x(3)=12	0x0000
0x0008	→	8	→	8
0x0010		10		10
0x0012		0x0012		8
		" 14		10
		" 16		12
		" 18		0x0018

Çok fazla bellek alanı harcanır. Gereksiz veri tutar.

Preallocation yaparsak;

```
x = zeros (1,3);   -> 1x3 matris
x = 8
x(2) = 10;
x(3) = 12;
```

8	→	8	→	8	olur.
0		10		10	
0		0		12	

try- catch

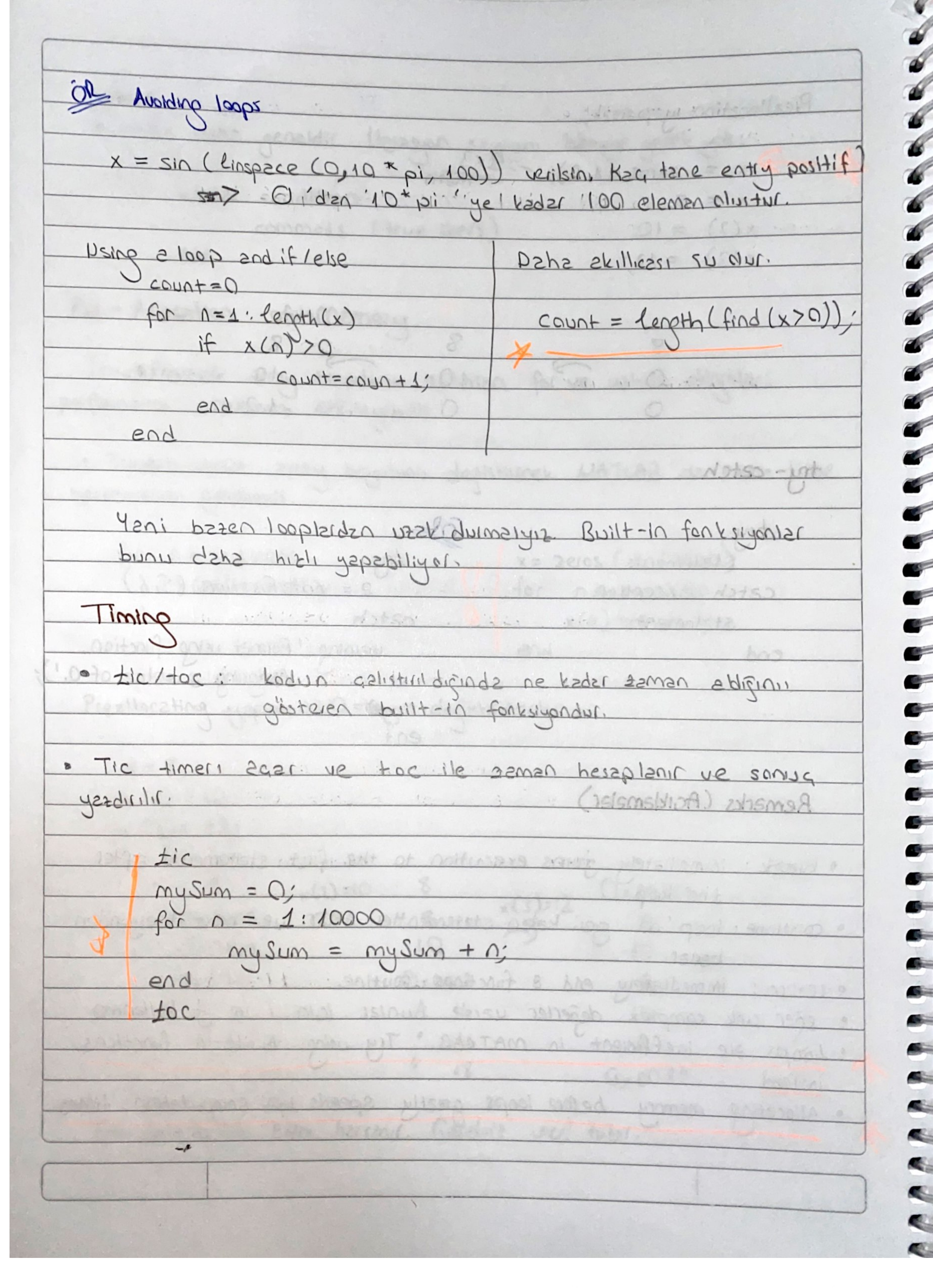
```
try
   statements
catch exception
   statements
end
```

OR

```
try
   a = notaFunction (5,6)
catch
   warning ('Format using function.
            Assigning a value of 0.');
   a = 0;
end
```

Remarks (Açıklamalar)

- break: immediately jums execution to the first statement after the loop.
- continue: loop'da geri kalan statementleri atlar ve next iterasyondan başlar.
- return: immediately end a functions routine.
- eğer çok complex değerler varsa bunlar için i ve j kullanma
- Loops are inefficient in MATLAB. Try using built-in functions instead
- Allocating memory before loops greatly speeds up computation times



ÖR Avoiding loops

x = sin (linspace (0,10 * pi, 100)) verilsin, Kaç tane entry pozitif?
=> 0'dan 10*pi'ye kadar 100 eleman oluştur.

Using a loop and if/else

```
count=0
for n=1:length(x)
    if x(n)>0
        count=coun+1;
    end
end
```

Daha akıllıcası şu olur.

```
count = length(find(x>0));
```

Yani bazen looplardan uzak durmalıyız. Built-in fonksiyonlar bunu daha hızlı yapabiliyor.

Timing

- tic/toc : kodun çalıştırıldığında ne kadar zaman aldığını gösteren built-in fonksiyondur.

- Tic timeri açar ve toc ile zaman hesaplanır ve sonuç yazdırılır.

```
tic
mySum = 0;
for n = 1:10000
    mySum = mySum + n;
end
toc
```

• MATLAB ayrıca Profiler ile kodların çalışma zamanında detaylı rapor oluşturur.

• Bunu ise profile on, profile off ve profile viewer metoduyla yaparız.

```
>> profile on
>> test_for
Elapsed time is 0.039972 seconds
>> profile viewer
>> profile off
```

Data Structures

• 2D' matrisler kullandık.
  - n boyutlu olabilir.
  - her eleman aynı tipte olmalı. (integers, doubles, characters...)
  - Matrisler space-efficient ve convenient for calculation.

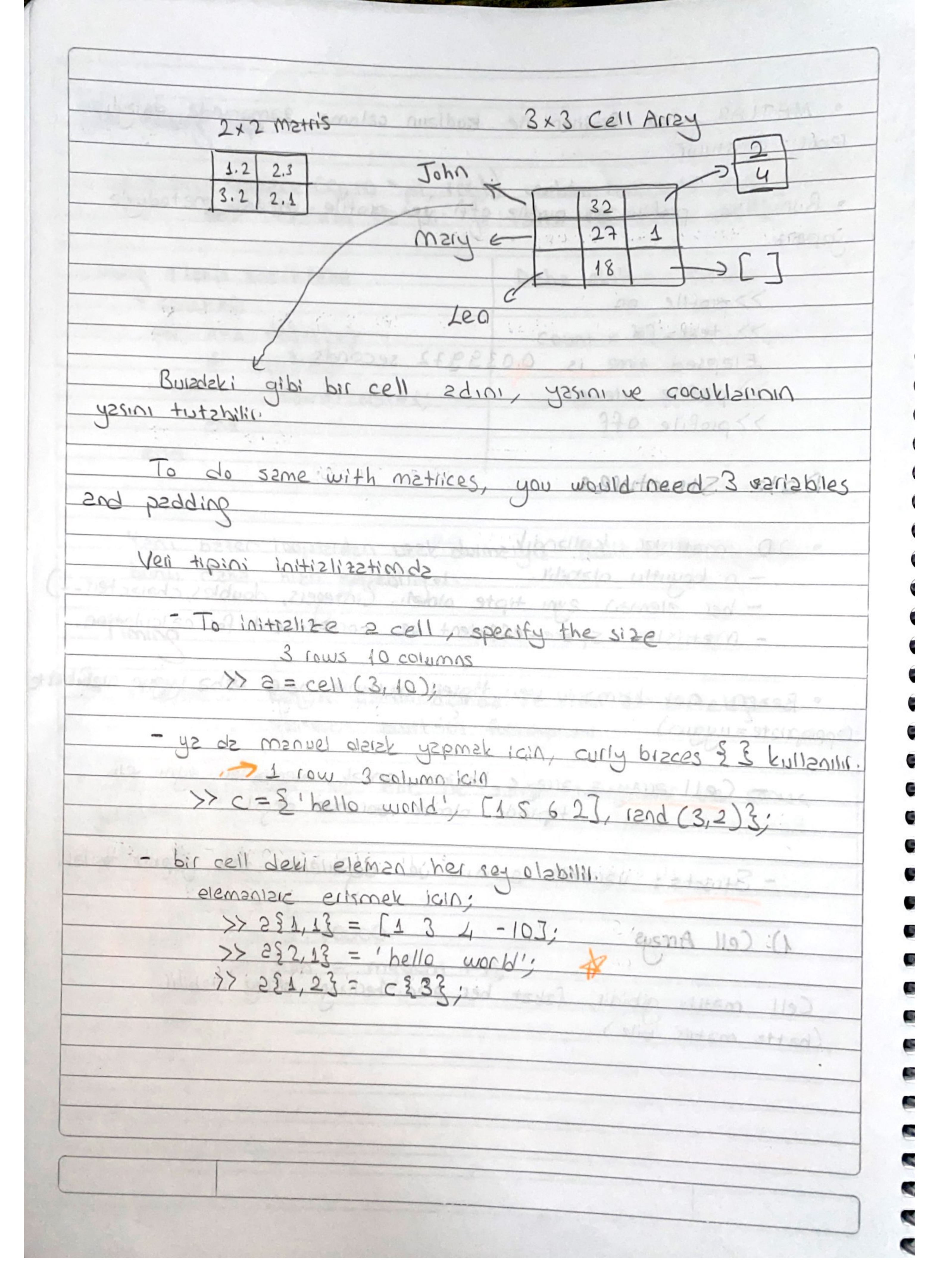
• Bazen çok karmaşık veri tipleri durumuna göre daha uygun olabilirler. (appropriate = uygun)

  - Cell array = array'e benzer, ancak elemanları aynı veri tipinde olmak zorunda değil.

  - Structs: variable adını ve değerlerini tek yapıda toplar.

1) Cell Arrays

Cell matrix gibidir, fakat her alan herhangi birşey alabilir. (hatta matris bile)



2x2 matris

1.2	2.3
3.2	2.1

3x3 Cell Array

Buradaki gibi bir cell adını, yasını ve cocuklarının yasını tutabilir.

To do same with matrices, you would need 3 variables and padding

Veri tipini initializationda

- To initialize a cell, specify the size

  3 rows 10 columns

```
>> a = cell(3,10);
```

- ya da manuel olarak yapmak icin, curly braces { } kullanılır.

  1 row 3 column icin

```
>> c = {'hello world', [1 5 6 2], rand(3,2)};
```

- bir cell deki eleman her şey olabilir.

  elemanlara erişmek icin;

```
>> a{1,1} = [1 3 4 -10];
>> a{2,1} = 'hello world';
>> a{1,2} = c{3};
```

relevant=ilgili.

2) Structs

- Structlar isim eklemenizi sağlayan ve ilgili değişkenleri saklar.
  - C' deki gibi objeler var.

- To initialize an empty struct

```
>> s = struct([]);
```

size(s) 1x1 olur.

- initialization is optional but is recommended when using large structs

- To add fields
  - Field değerleri herhangi bir şey olabilir: matrix, cell hatta struct.
  - useful for keeping variables together

```
>> s.name = 'Aslı Hazım';
>> s.scores = [80, 60, 70];
>> s.status = 'P';
```

↳ substring kullan s(1).adi - gibi

cell2mat - cell2struct

- cell2mat => convert cell array to ordinary array of the underlying data type.

```
>> C = {[1], [2,3,4]; [5; 9], [6 7 8; 10 11 12]}
```

```
C =
    2x2 cell array

{[ 1 ]}  {1x3 double}
{2x1 double} {2x3 double}
```

```
>> A = cell2mat(C)
```

olur.

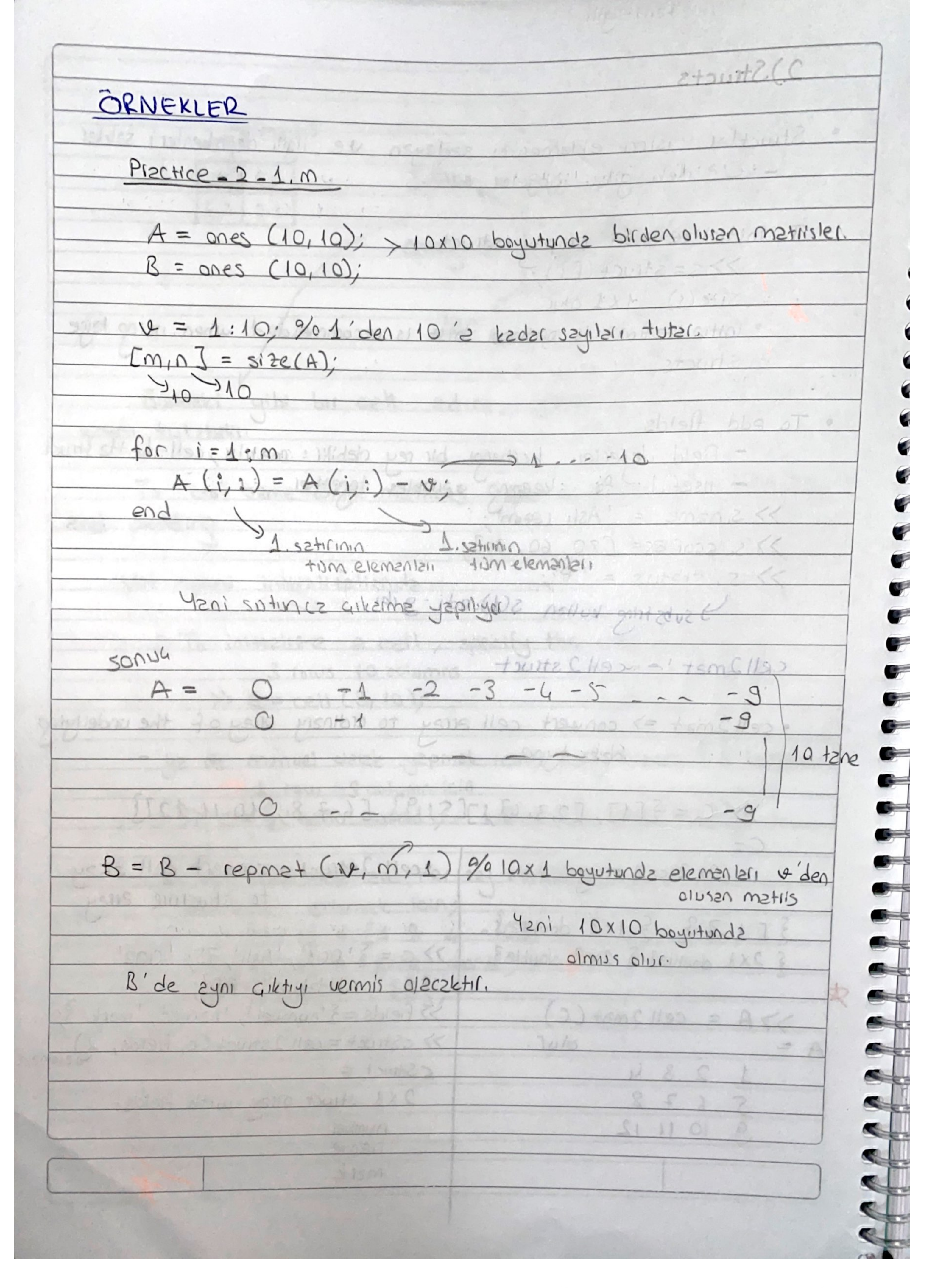
```
A =
    1  2  3  4
    5  6  7  8
    9  10 11 12
```

- cell2struct = convert cell array to structure array.

```
>> c = {'001', 'Aslı', 75; '002'
        'Hakim', 85};
>> fields = {'number', 'name', 'mark'};
>> cStruct = cell2struct(c, fields, 2)
```

↳ eleman

```
cStruct =
    2x1 struct array with fields:
        number
        name
        mark
```



ÖRNEKLER

Practice_2_1.m

```
A = ones (10,10);   > 10x10 boyutunda birden oluşan matrisler.
B = ones (10,10);

v = 1:10; % 1'den 10'a kadar sayıları tutar.
[m,n] = size(A);
  ↳10  ↳10

for i = 1:m
   A(i,:) = A(i,:) - v;      → 1 ... 10
end
```

A(i,:) → 1. satırının tüm elemanları — A(i,:) → 1. satırının tüm elemanları

Yani satırınca çıkarma yapılıyor.

sonuç

```
A =  0  -1  -2  -3  -4  -5  ...  -9
     0  -1                       -9
                                        } 10 tane
     0  -1                       -9
```

```
B = B - repmat (v, m, 1)   % 10x1 boyutunda elemanları v'den oluşan matris
```

Yani 10x10 boyutunda olmuş olur.

B'de aynı çıktıyı vermiş olacaktır.

CamScanner ile tarandı

Practice-2-2.m

```
A = randi ([-10,10], 10, 10);     10x10 boyutunda elemanlar
                                  -10 ile 10 arasında matris.

[m,n] = size(A);

B = zeros (m,n);
C = zeros (m,n);
%% A'daki pozitif elemanları B'ye atalım
section anlamı

for i = 1:m
    for j = 1:n
        if A(i,j) > 0
            B(i,j) = A(i,j);
        end
    end
end
                    Run section diyerek burası
                    çalıştırılabilir.

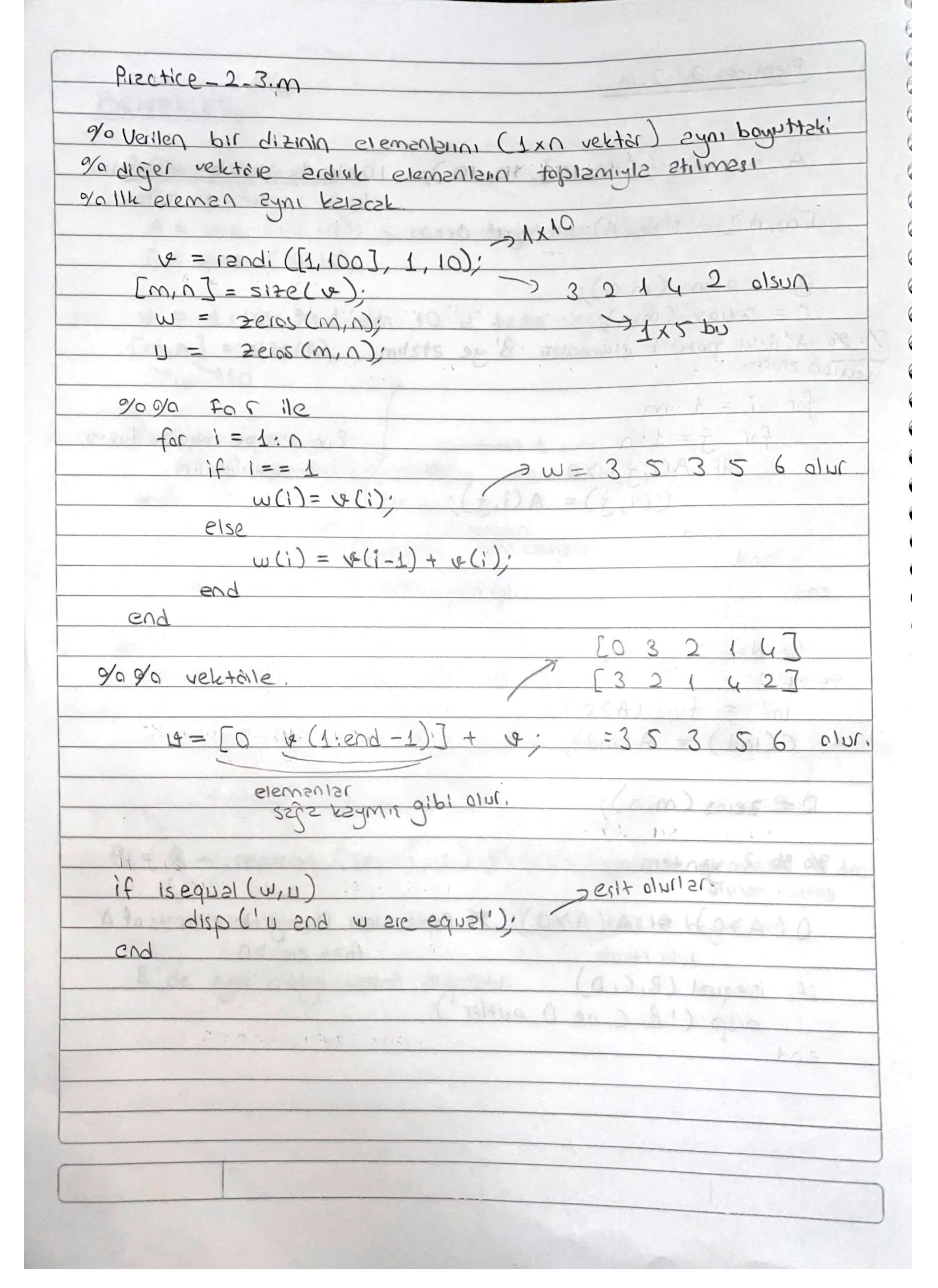
Ya da 2. yöntem
%% 2.
ind = find (A>0);
C(ind) = A(ind);

D = zeros (m,n);

%% 3. yöntem

D(A>0) = A(A>0); % Copies into D only the elements of A
                   that are >0

if isequal (B,C,D)
    disp ('B, C ve D esitler')
end
```



Practice_2_3.m

```
% Verilen bir dizinin elemanlarını (1xn vektör) aynı boyuttaki
% diğer vektöre ardışık elemanlarının toplamıyla atılması
% ilk eleman aynı kalacak.

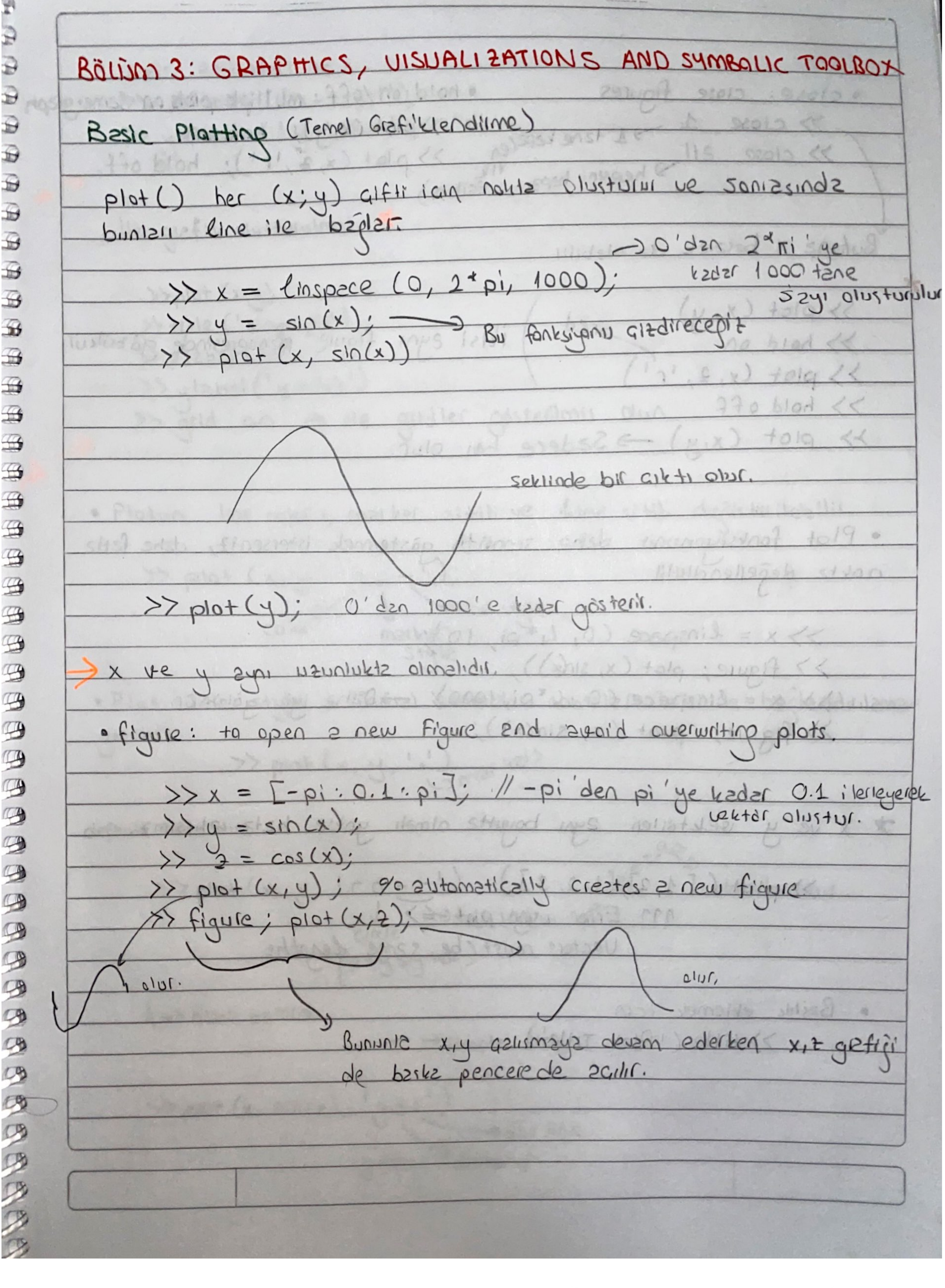
    v = randi([1,100], 1, 10);        → 1x10
    [m,n] = size(v);                  → 3 2 1 4 2 olsun
    w = zeros(m,n);                   → 1x5 bu
    u = zeros(m,n);

  %% for ile
    for i = 1:n
        if i == 1
            w(i) = v(i);              → w = 3 5 3 5 6 olur
        else
            w(i) = v(i-1) + v(i);
        end
    end

  %% vektörle                         [0 3 2 1 4]
                                      [3 2 1 4 2]
    u = [0  v(1:end-1)] + v;          = 3 5 3 5 6 olur.

        elemanlar sağa kaymış gibi olur.

  if isequal(w,u)
      disp('u and w are equal');      → eşit olurlar.
  end
```



BÖLÜM 3: GRAPHICS, VISUALIZATIONS AND SYMBALIC TOOLBOX

Basic Platting (Temel Grafiklendirme)

plot() her (x;y) çifti için nokta oluşturur ve sonrasında bunları line ile bağlar.

```
>> x = linspace (0, 2*pi, 1000);
>> y = sin(x);
>> plot (x, sin(x))
```

→ 0'dan 2*pi'ye kadar 1000 tane sayı oluşturulur

y = sin(x) → Bu fonksiyonu çizdireceğiz

şeklinde bir çıktı olur.

```
>> plot(y);
```

0'dan 1000'e kadar gösterir.

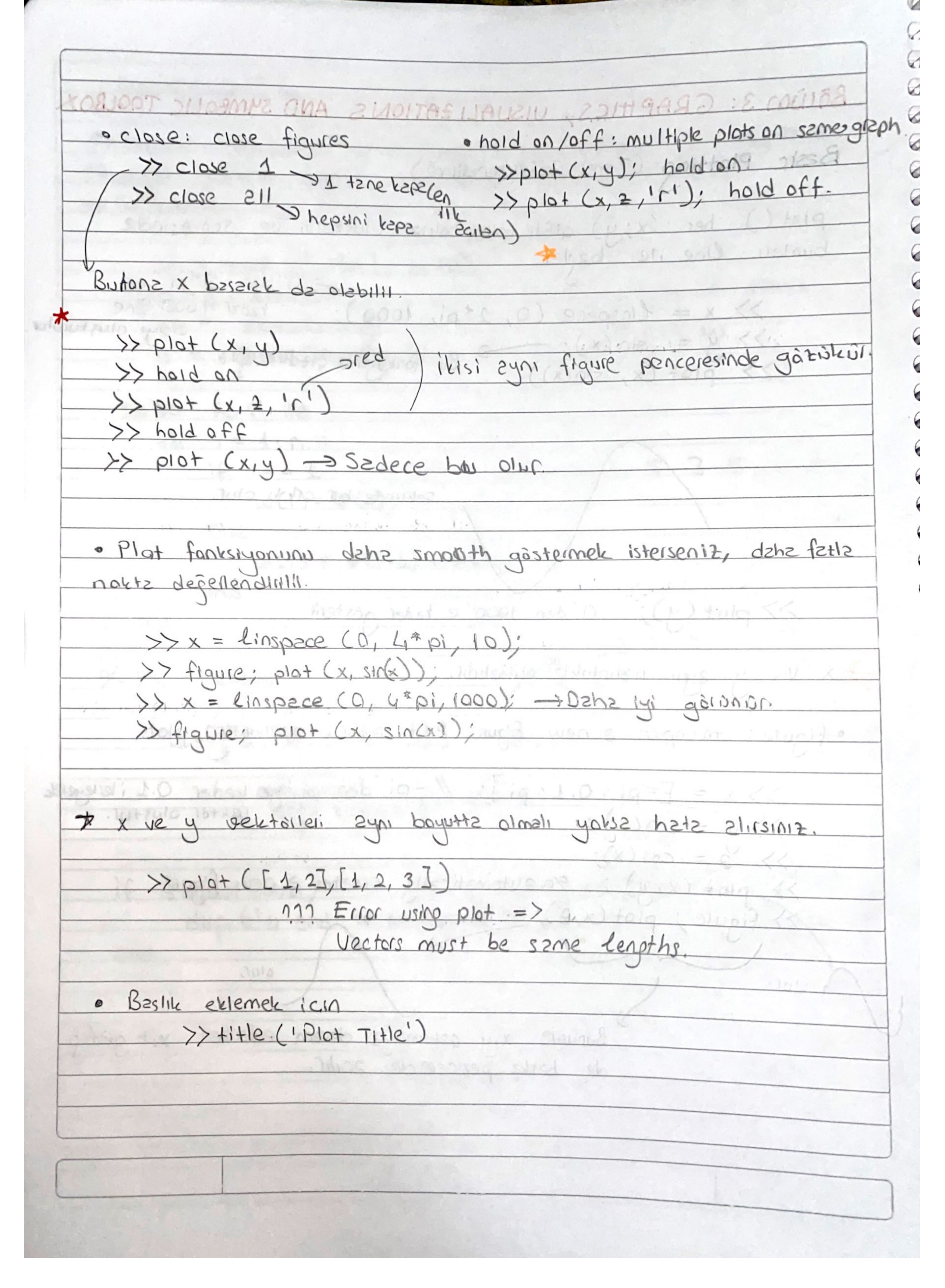
→ x ve y aynı uzunlukta olmalıdır.

- figure: to open a new figure and avoid overwriting plots.

```
>> x = [-pi : 0.1 : pi];   // -pi'den pi'ye kadar 0.1 ilerleyerek vektör oluştur.
>> y = sin(x);
>> z = cos(x);
>> plot (x,y);   % automatically creates a new figure
>> figure; plot (x,z);
```

olur. olur.

Bununla x,y çalışmaya devam ederken x,z grafiği de başka pencerede açılır.



• close: close figures

```
>> close 1    → 1 tane kapa(ilk açılan)
>> close all  → hepsini kapa
```

Butona x basarak da olabilir.

• hold on/off: multiple plots on same graph

```
>>plot (x,y); hold on
>> plot (x, z, 'r'); hold off.
```

*

```
>> plot (x, y)
>> hold on
>> plot (x, z, 'r')    (red)
>> hold off
>> plot (x,y)  → Sadece bu olur
```

(plot(x, y), hold on, plot(x, z, 'r')) ikisi aynı figure penceresinde gözükür.

• Plot fonksiyonunu daha smooth göstermek isterseniz, daha fazla nokta değerlendirilir.

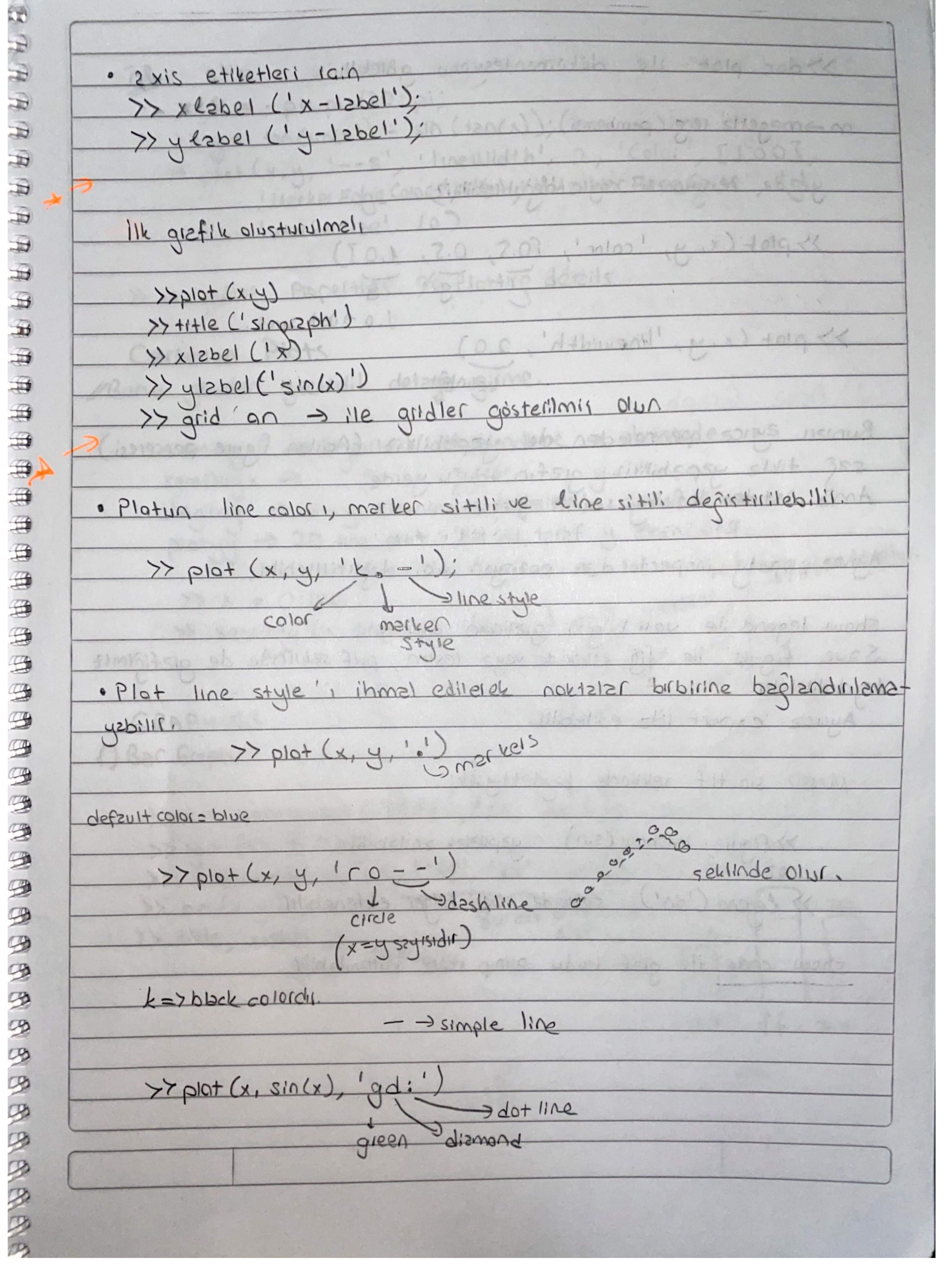
```
>> x = linspace (0, 4*pi, 10);
>> figure; plot (x, sin(x));
>> x = linspace (0, 4*pi, 1000);  → Daha iyi görünür
>> figure; plot (x, sin(x));
```

* x ve y vektörleri aynı boyutta olmalı yoksa hata alırsınız.

```
>> plot ([1, 2], [1, 2, 3])
??? Error using plot =>
Vectors must be same lengths.
```

• Başlık eklemek için

```
>> title ('Plot Title')
```


• axis etiketleri için
>> xlabel ('x-label');
>> ylabel ('y-label');
Ilk grafik oluşturulmalı
>>plot (x,y)
>> title ('singraph')
>> xlabel ('x')
>> ylabel ('sin(x)')
>> grid on → ile gridler gösterilmiş olur
• Plotun line colorı, marker sitili ve line sitili değiştirilebilir.
>> plot (x, y, 'k.-');
color
marker style
line style
• Plot line style'ı ihmal edilerek noktalar birbirine bağlandırılamayabilir.
>> plot (x, y, '.')
markers
default color = blue
>> plot (x, y, 'ro--')
circle
(x=y sayısıdır)
dash line
şeklinde olur.
k => black colordır.
— → simple line
>> plot (x, sin(x), 'gd:')
green
diamond
dot line

>>doc plot ile dokümantasyonu görebiliriz.

m→magenta rengi (pembemsi)

ye de rengimizi şöyle değiştirebiliriz;

>>plot (x, y, 'color', [0.5, 0.5, 1.0])

(0.5 → r, 0.5 → g, 1.0 → b)

1.0 → üst sınır.

>>plot (x, y, 'linewidth', 2.0)

(2.0 → kalınlığı)

Bunları ayrıca pencereden de yapabiliriz. (Açılan figure penceresi)
sağ tıkla yapabiliriz. grafın olduğu yerde.
Ama üstten fareyi seç ilk başta

Ayrıca property inspector'dan pozisyon vb. değiştirilebilir.

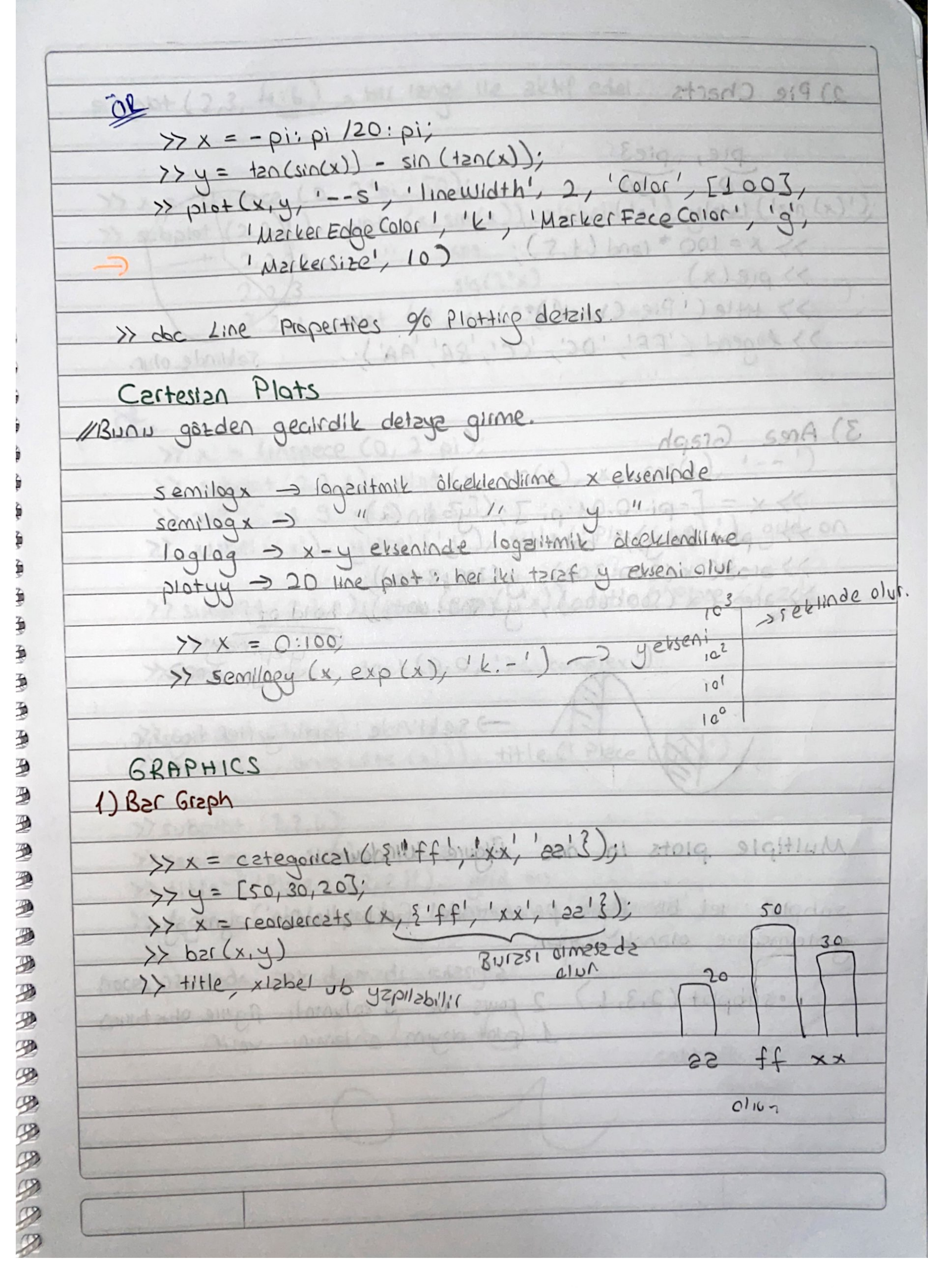
show legend ile veri bilgisi grafikte gösterilir.
Save figure ile .fig şeklinde veya resim, pdf şeklinde de grafiğimiz kaydedilebilir.
Ayrıca export da edilebilir.

Mesela sin.tif şeklinde kaydettiysek;

>> figure; imshow (sin) yapılarak açılabilir.

>> legend ('sin') şeklinde veri bilgisi eklenebilir.

show code ile graf kodu alınıp sonra kullanılabilir.



ÖR

```
>> x = -pi: pi /20: pi;
>> y = tan(sin(x)) - sin(tan(x));
>> plot(x,y, '--s', 'lineWidth', 2, 'Color', [1 0 0],
        'MarkerEdgeColor', 'k', 'MarkerFaceColor', 'g',
        'MarkerSize', 10)
```

>> doc Line Properties % Plotting details

Cartesian Plots

//Bunu gözden geçirdik detaya girme.

semilogx → logaritmik ölçeklendirme x ekseninde
semilogx → " " y "
loglog → x-y ekseninde logaritmik ölçeklendirme
plotyy → 2D line plot : her iki taraf y ekseni olur.

```
>> x = 0:100;
>> semilogy(x, exp(x), 'k.-') → y ekseni
```

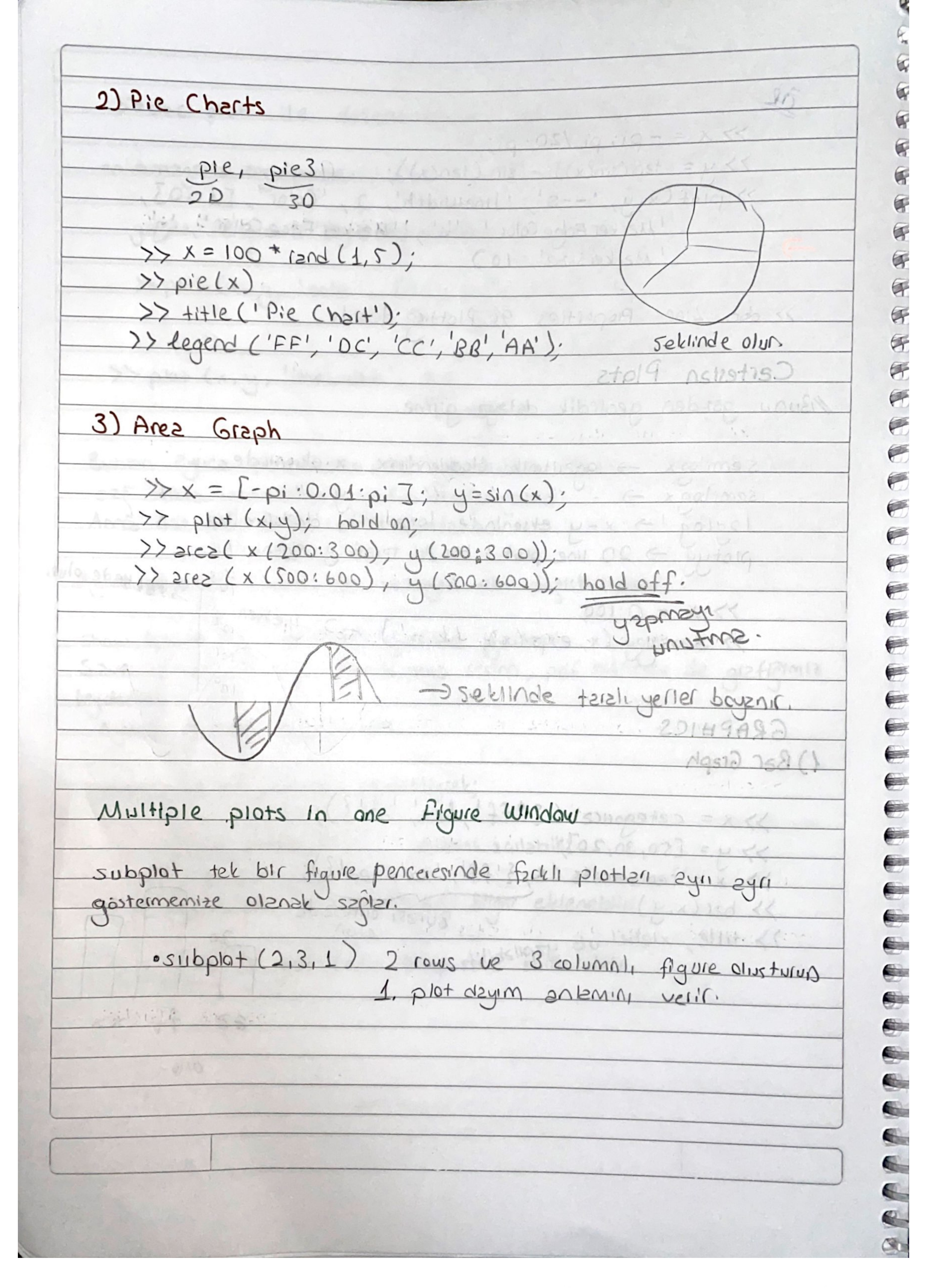
10^3, 10^2, 10^1, 10^0 → şeklinde olur.

GRAPHICS

1) Bar Graph

```
>> x = categorical({'ff', 'xx', 'aa'});
>> y = [50,30,20];
>> x = reordercats(x, {'ff', 'xx', 'aa'});
>> bar(x,y)
>> title, xlabel vb. yapılabilir
```

Burası olmasada olur



2) Pie Charts

pie, pie3
2D   3D

```
>> x = 100 * rand(1,5);
>> pie(x)
>> title('Pie Chart');
>> legend('FF', 'DC', 'CC', 'BB', 'AA');
```

şeklinde olur

3) Area Graph

```
>> x = [-pi:0.01:pi]; y=sin(x);
>> plot(x,y); hold on;
>> area(x(200:300), y(200:300));
>> area(x(500:600), y(500:600)); hold off.
```

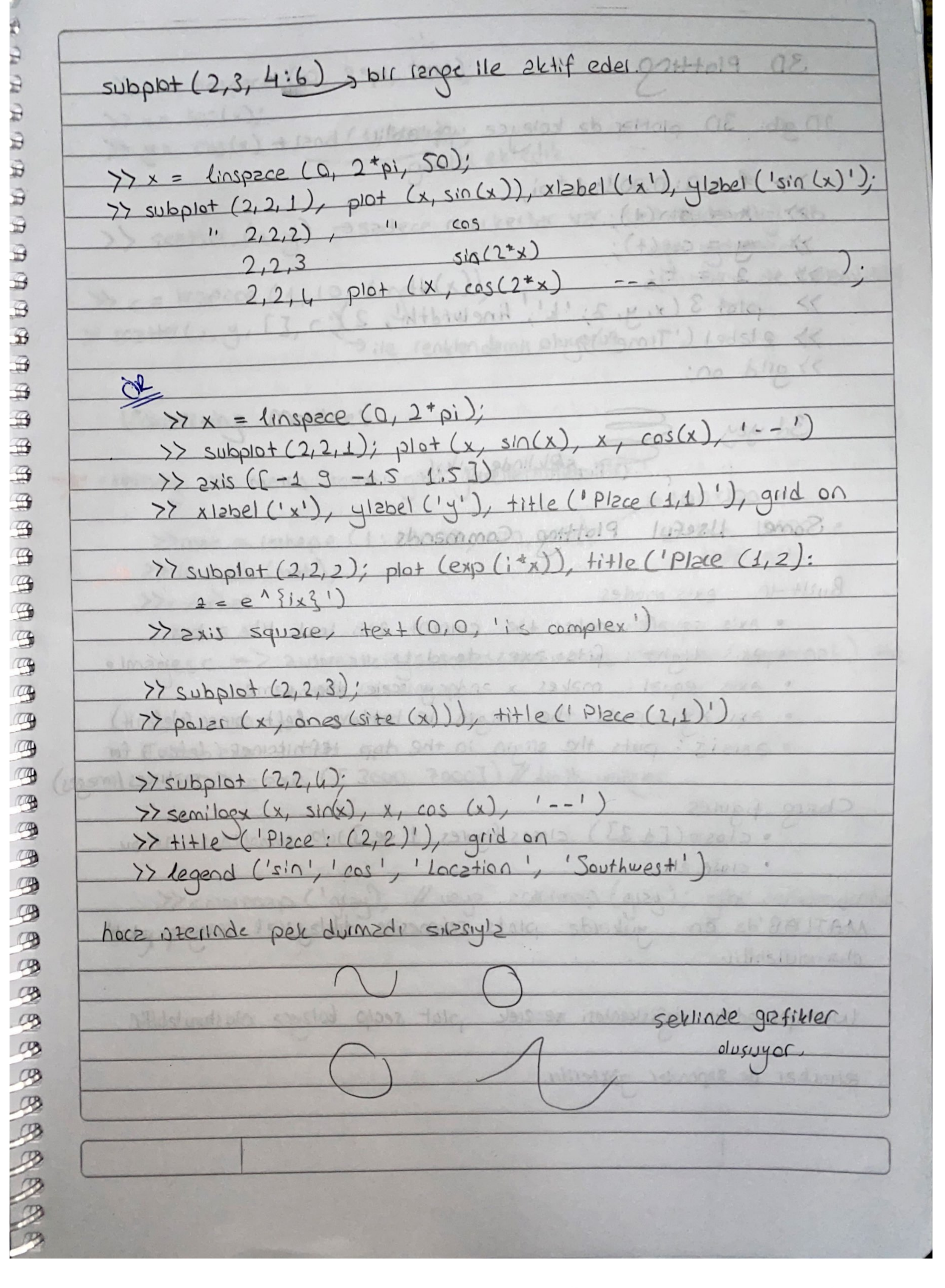
hold off. yapmayı unutma.

→ şeklinde taralı yerler boyanır.

Multiple plots in one Figure Window

subplot tek bir figure penceresinde farklı plotları ayrı ayrı göstermemize olanak sağlar.

- subplot(2,3,1) 2 rows ve 3 columnlı figure oluşturur 1. plot dizayn anlamını verir.



subplot(2,3,4:6) → bir range ile aktif eder.

```
>> x = linspace(0, 2*pi, 50);
>> subplot(2,2,1), plot(x, sin(x)), xlabel('x'), ylabel('sin(x)');
       "   2,2,2) ,    "          cos
           2,2,3                  sin(2*x)
           2,2,4   plot(x, cos(2*x)        ---          );
```

Ör

```
>> x = linspace(0, 2*pi);
>> subplot(2,2,1); plot(x, sin(x), x, cos(x), '--')
>> axis([-1 9 -1.5 1.5])
>> xlabel('x'), ylabel('y'), title('Place (1,1)'), grid on

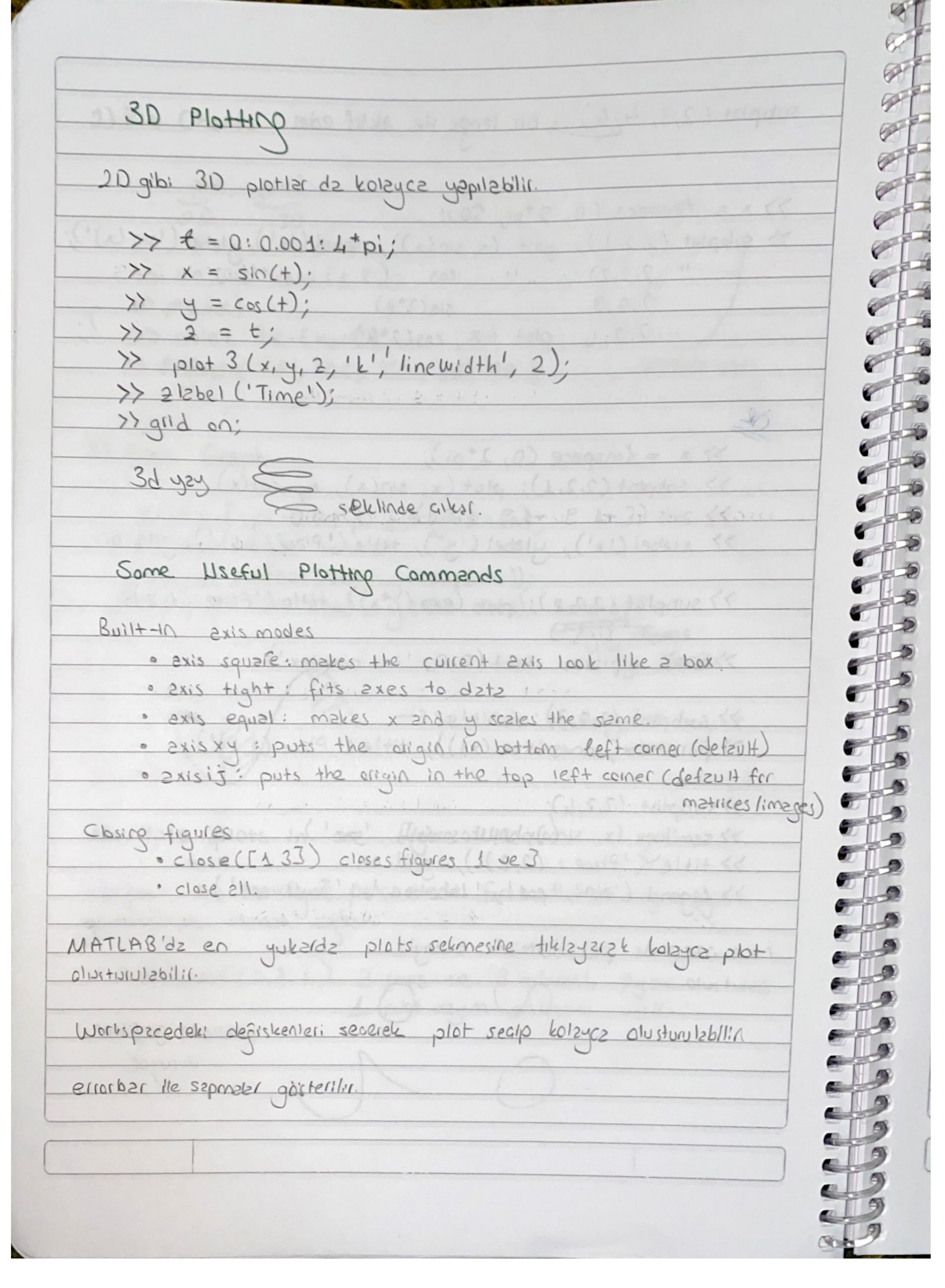
>> subplot(2,2,2); plot(exp(i*x)), title('Place (1,2):
   z = e^{ix}')
>> axis square, text(0,0,'is complex')

>> subplot(2,2,3);
>> polar(x, ones(size(x))), title('Place (2,1)')

>> subplot(2,2,4);
>> semilogx(x, sin(x), x, cos(x), '--')
>> title('Place: (2,2)'), grid on
>> legend('sin', 'cos', 'Location', 'Southwest')
```

hoca üzerinde pek durmadı sırasıyla

şeklinde grafikler oluşuyor.



3D Plotting

2D gibi 3D plotlar da kolayca yapılabilir.

```
>> t = 0:0.001:4*pi;
>> x = sin(t);
>> y = cos(t);
>> z = t;
>> plot3(x, y, z, 'k', 'linewidth', 2);
>> zlabel('Time');
>> grid on;
```

3d yay şeklinde çıkar.

Some Useful Plotting Commands

Built-in axis modes

- axis square: makes the current axis look like a box.
- axis tight: fits axes to data
- axis equal: makes x and y scales the same.
- axis xy: puts the origin in bottom left corner (default)
- axis ij: puts the origin in the top left corner (default for matrices / images)

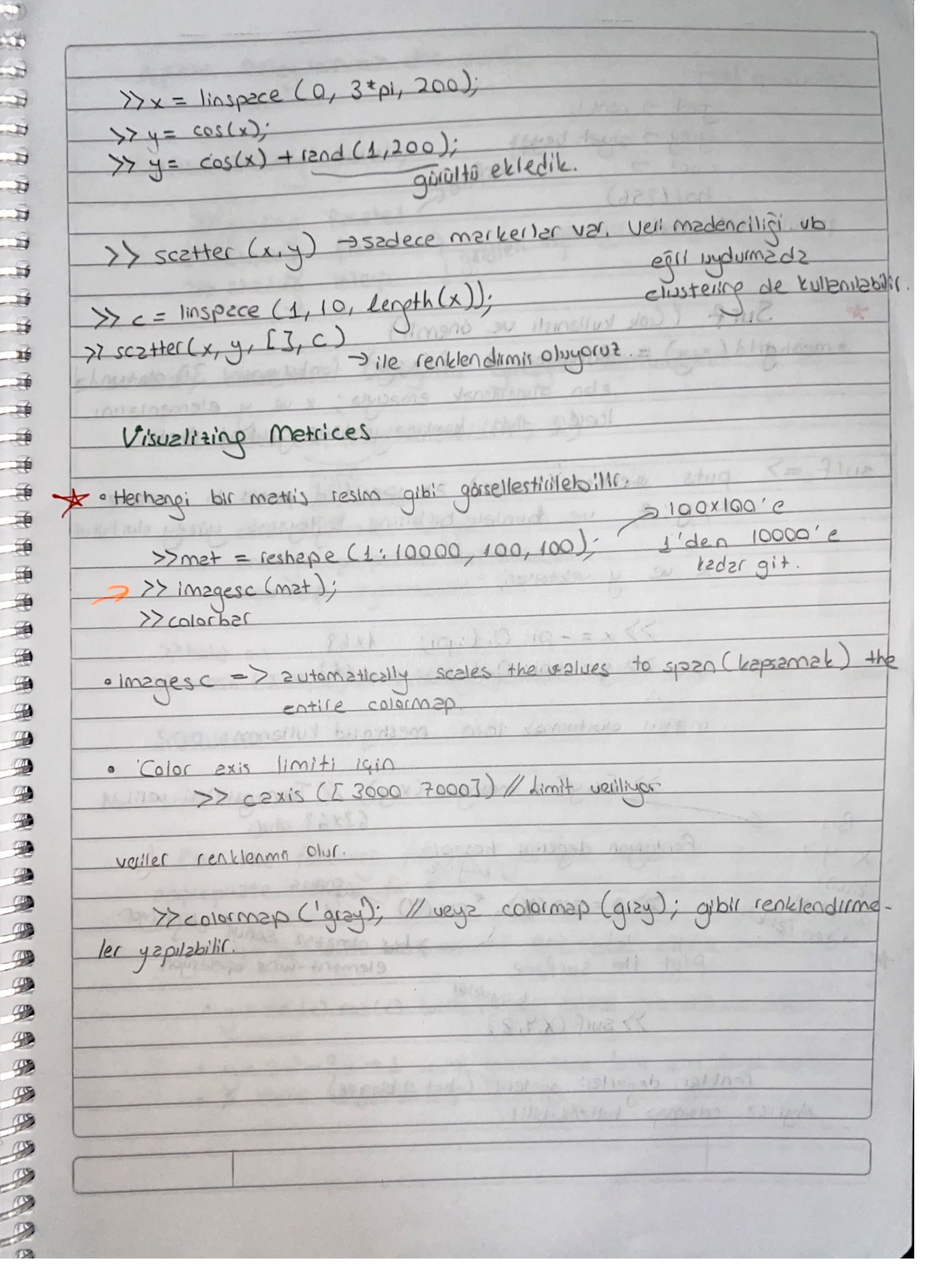
Closing figures

- close([1 3]) closes figures 1 ve 3
- close all.

MATLAB'da en yukarıda plots sekmesine tıklayarak kolayca plot oluşturulabilir.

Workspacedeki değişkenleri seçerek plot seçip kolayca oluşturulabilir.

errorbar ile sapmalar gösterilir.



```
>> x = linspace (0, 3*pi, 200);
>> y = cos(x);
>> y = cos(x) + rand (1,200);
```
gürültü ekledik.

```
>> scatter (x,y)
```
→ sadece markerlar var. Veri madenciliği vb eğri uydurmada clustering de kullanılabilir.

```
>> c = linspace (1, 10, length(x));
>> scatter(x, y, [], c)
```
→ ile renklendirmiş oluyoruz.

Visualizing Metrices

- Herhangi bir matris resim gibi görselleştirilebilir.

```
>> mat = reshape (1:10000, 100, 100);
>> imagesc (mat);
>> colorbar
```
→ 100x100'e 1'den 10000'e kadar git.

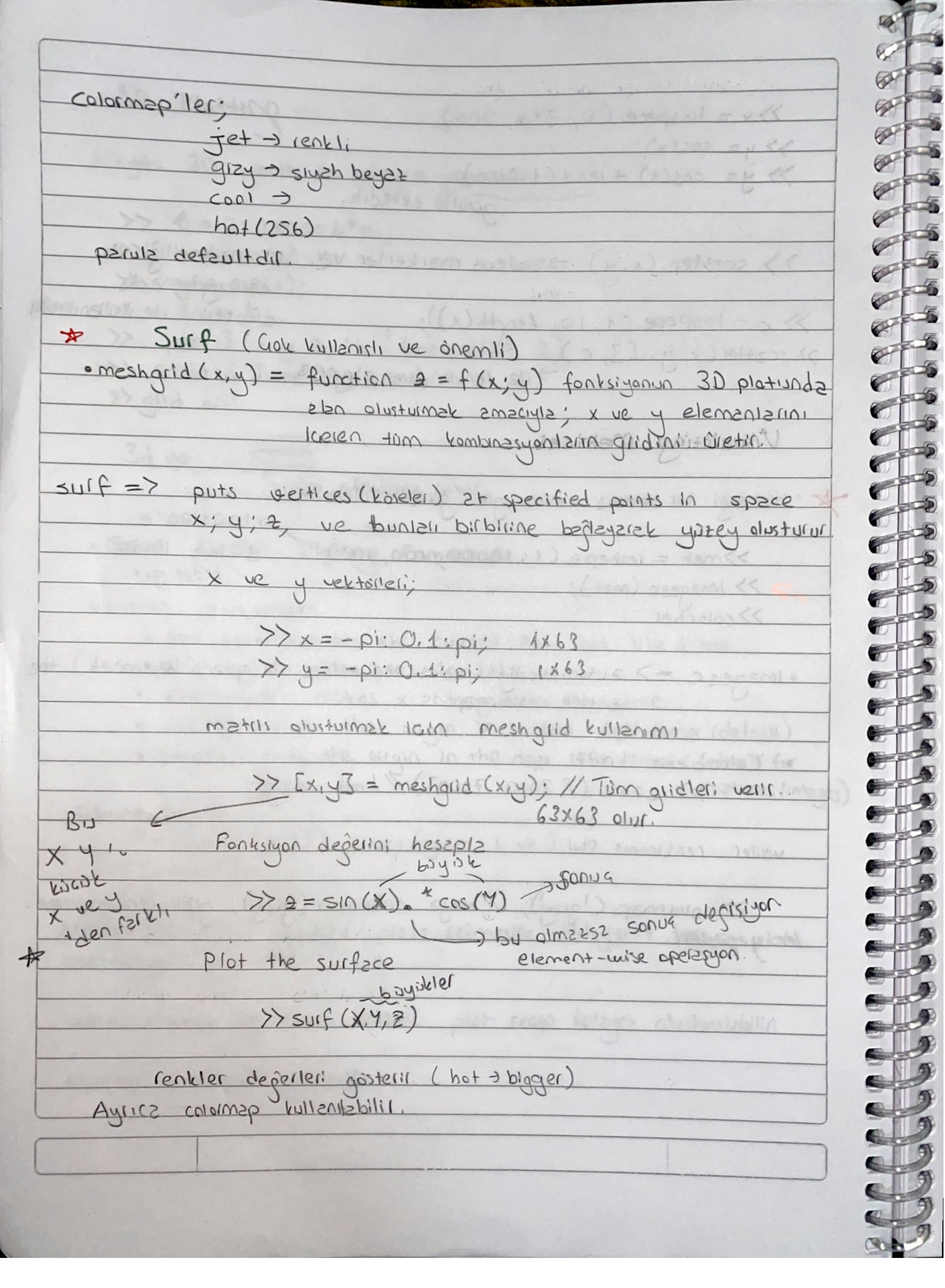
- imagesc => automatically scales the values to span (kapsamak) the entire colormap.

- Color axis limiti için
```
>> caxis ([3000 7000]) // limit veriliyor
```

veriler renklenmiş olur.

```
>> colormap ('gray'); // veya colormap (gray); gibi
```
renklendirmeler yapılabilir.

CamScanner ile tarandı



colormap'ler;

jet → renkli
gray → siyah beyaz
cool →
hot(256)

parula defaultdır.

★ Surf (Çok kullanışlı ve önemli)

• meshgrid(x,y) = function z = f(x;y) fonksiyonun 3D platunda alan oluşturmak amacıyla; x ve y elemanlarını içeren tüm kombinasyonların gridini üretir.

surf => puts vertices (köşeler) at specified points in space x; y; z, ve bunları birbirine bağlayarak yüzey oluşturur

x ve y vektörleri;

```
>> x = -pi: 0.1: pi;     1x63
>> y = -pi: 0.1: pi;     1x63
```

matris oluşturmak için meshgrid kullanımı

```
>> [x,y] = meshgrid(x,y); // Tüm gridleri verir.
```

63x63 olur.

Bu X Y'ler küçük x ve y'den farklı

Fonksiyon değerini hesapla

```
>> z = sin(X).*cos(Y)
```

büyük → sonuç

bu olmazsa sonuç değişiyor element-wise operasyon.

★ Plot the surface

```
>> surf(X,Y,Z)
```

büyükler

renkler değerleri gösterir (hot → bigger)

Ayrıca colormap kullanılabilir.

Ayrıntılı Detay için => doc surf.

$a^2 + b^2 = r^2$

r.cos

Surface Shading türleri;

```
>> shading faceted  -> default
>> shading flat
>> shading interp
```

} gridleri değiştiriyor (smoother)

contour; kullanılarak 2 Boyutlu yüzeyler oluşturulabilir.

```
>> contour(x, y, z, 'LineWidth', 2)
```

(x, y, z) surf argümanları

color indicates height
linestyle properties değiştirilebilir.
colormap uygulanabilir.

```
>> hold on
>> mesh(x, y, z)
```

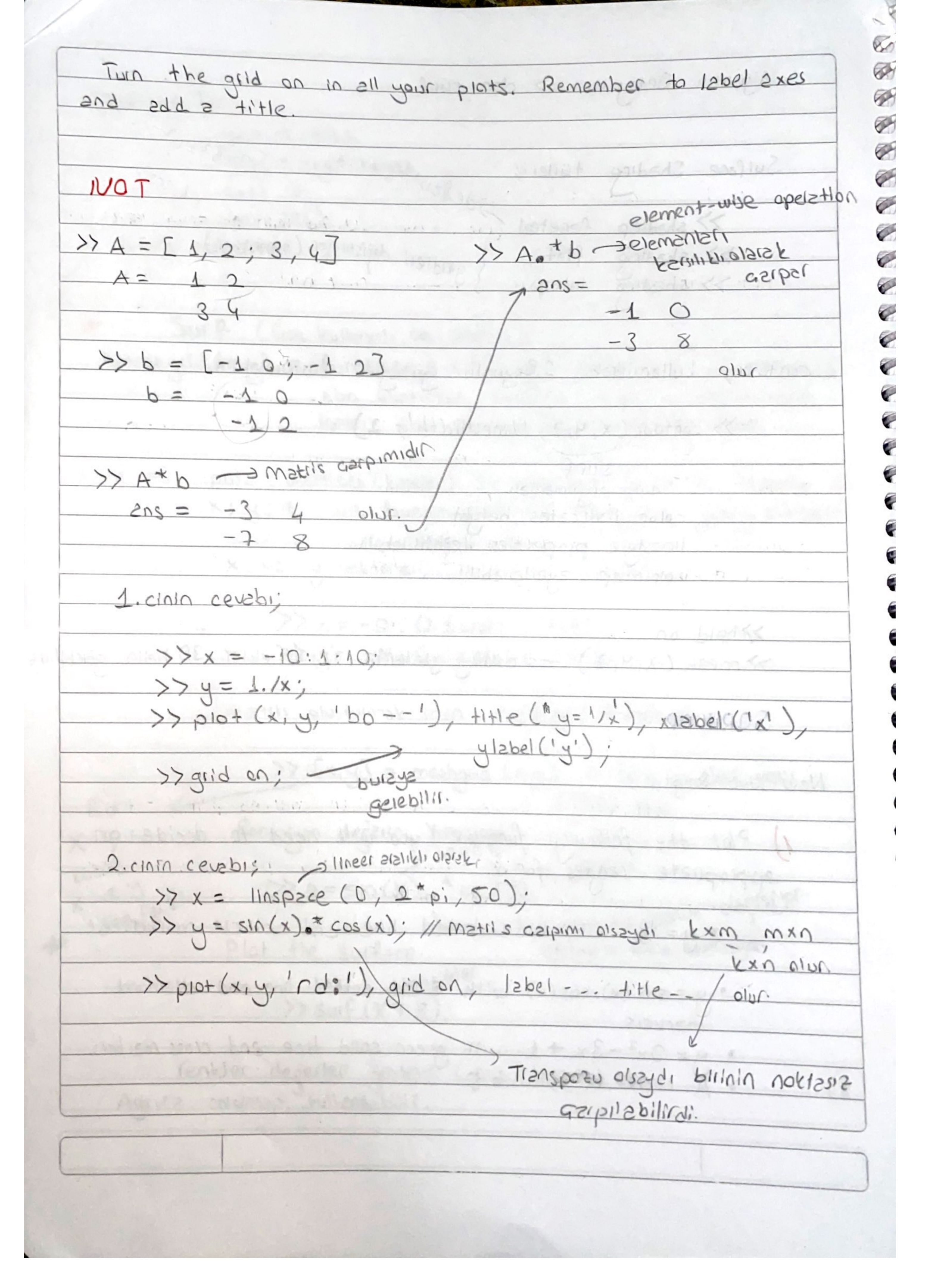
-> detay gösterilir. üstüne ekler 3D halini görebiliriz

SORULAR

New Live Script.

1) Plot the following functions (you will need to decide on appropriate ranges for x

- $y = 1/x$ with blue dashed line and circle markers
- $y = \sin(x).\cos(x)$ with a red dotted line and diamond markers
- $y = 2x^2 - 3x + 1$ with green solid line and cross markers.
- A circle of radius = 5



Turn the grid on in all your plots. Remember to label axes and add a title.

NOT

```
>> A = [1, 2; 3, 4]
A =   1  2
      3  4

>> b = [-1 0; -1 2]
b =   -1  0
      -1  2

>> A*b          → Matris çarpımıdır
ans =   -3   4     olur.
        -7   8
```

```
>> A.*b   → element-wise operation
ans =       elemanları karşılıklı olarak çarpar
      -1  0
      -3  8
            olur
```

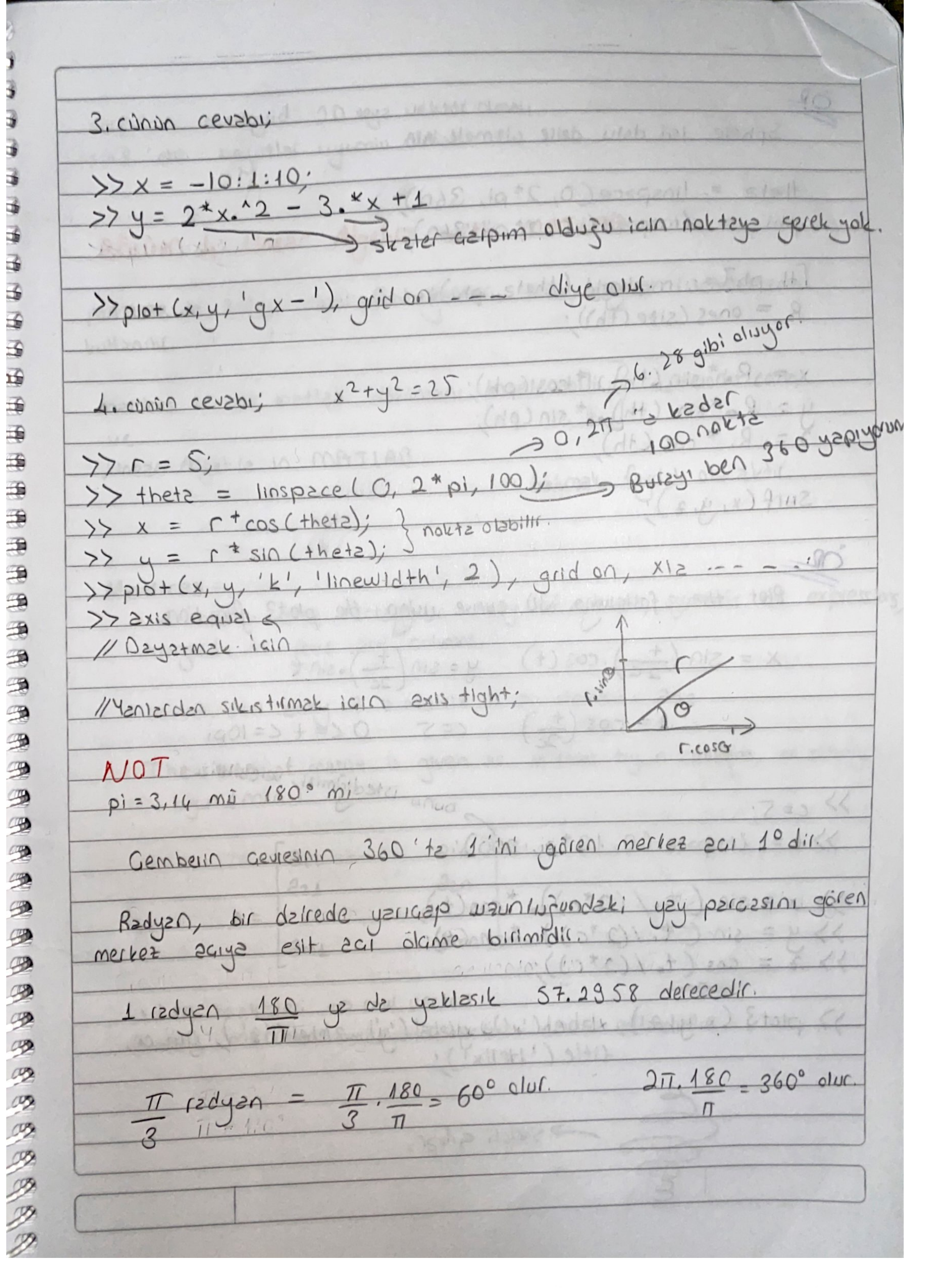
1. cinin cevabı;

```
>> x = -10:1:10;
>> y = 1./x;
>> plot(x, y, 'bo--'), title('y=1/x'), xlabel('x'),
   ylabel('y');
>> grid on;          → buraya gelebilir.
```

2. cinin cevabı;

```
>> x = linspace(0, 2*pi, 50);        → lineer aralıklı olarak
>> y = sin(x).*cos(x);  // matris çarpımı olsaydı  kxm , mxn
                                                    kxn olur
>> plot(x,y,'rd:'), grid on, label ... title ...   olur
```

Transpozu olsaydı birinin noktasız çarpılabilirdi.



3. cunun cevabı;

```
>> x = -10:1:10;
>> y = 2*x.^2 - 3.*x + 1
```

→ skaler çarpım olduğu için noktaya gerek yok.

```
>> plot(x, y, 'gx-'), grid on
```

--- diye olur.

4. cunun cevabı; $x^2+y^2=25$

```
>> r = 5;
>> theta = linspace(0, 2*pi, 100);
>> x = r*cos(theta);
>> y = r*sin(theta);
>> plot(x, y, 'k', 'linewidth', 2), grid on, xla ---
>> axis equal
```

→ 0, 2π → 6.28 gibi alıyor. kadar 100 nokta

→ Burayı ben 360 yapıyorum

} nokta olabilir.

// Dayatmak için

// Yanlardan sıkıştırmak için axis tight;

NOT

pi = 3,14 mü 180° mi

Çemberin çevresinin 360'ta 1'ini gören merkez açı 1° dir.

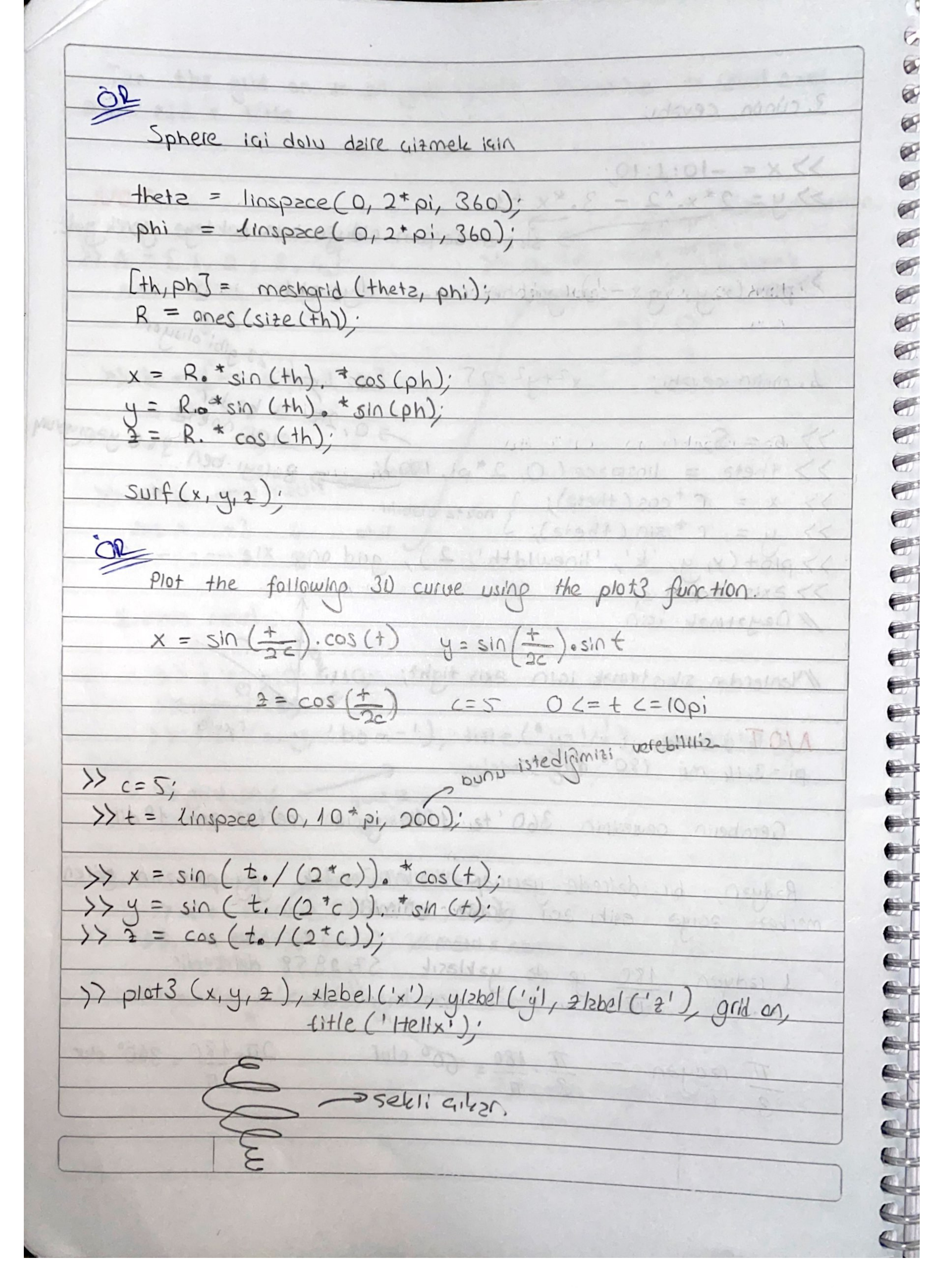
Radyan, bir dairede yarıçap uzunluğundaki yay parçasını gören merkez açıya eşit açı ölçme birimidir.

1 radyan $\frac{180}{\pi}$ ya da yaklaşık 57.2958 derecedir.

$\frac{\pi}{3}$ radyan $= \frac{\pi}{3} \cdot \frac{180}{\pi} = 60°$ olur.

$2\pi \cdot \frac{180}{\pi} = 360°$ olur.

CamScanner ile tarandı



ÖR

Sphere içi dolu daire çizmek için

```
theta = linspace(0, 2*pi, 360);
phi = linspace(0, 2*pi, 360);

[th, ph] = meshgrid(theta, phi);
R = ones(size(th));

x = R.*sin(th).*cos(ph);
y = R.*sin(th).*sin(ph);
z = R.*cos(th);

surf(x, y, z);
```

ÖR

Plot the following 3D curve using the plot3 function.

$x = \sin\left(\frac{t}{2c}\right) \cdot \cos(t)$ $\quad y = \sin\left(\frac{t}{2c}\right) \cdot \sin t$

$z = \cos\left(\frac{t}{2c}\right)$ $\quad c = 5$ $\quad 0 <= t <= 10pi$

```
>> c = 5;
>> t = linspace(0, 10*pi, 200);
```
(bunu istediğimizi verebiliriz → 200)

```
>> x = sin(t./(2*c)).*cos(t);
>> y = sin(t./(2*c)).*sin(t);
>> z = cos(t./(2*c));

>> plot3(x, y, z), xlabel('x'), ylabel('y'), zlabel('z'), grid on,
       title('Hellix');
```

→ şekli çıkar.

surf için meshgrid 2D veya vektör olmalı.
surf 'de boyutlar uyumlu olmalı.

BÖLÜM 4: Lineer Algebra with MATLAB

MATLAB'de lineer cebir fonksiyonları hızlı, sayısal işlemler için kullanılır.

matrices in the matlab dökümanı incelenebilir, } mathworks.com
ve
Lineer Algebra in MATLAB
bakmak faydalı olacaktır.

Matrices

- A matrix is a rectangular array of numbers, symbols or expressions arranged in rows and columns.
- an individual entry of a matrix is an element: a_{23}
- The size of matrix is given as m rows by n columns, or simply m by n (m×n)

$$A = \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ a_{21} & & a_{2n} \\ & & \\ a_{m1} & & a_{mn} \end{bmatrix}_{m \times n}$$

- 1×n matrisleri → row vectors
- m×1 matrisleri → column vectors olarak adlandırılır.

Matrix Dimensions

- MATLAB has several functions to determine the number of elements in and the dimensions of variables:

  - numel (vec) returns the number of elements in the vector.
  - numel (mat) returns the total number of elements in the matrix (the product of the number of rows and the number of columns.
  - length (vec) returns the number of elements in the vector.
  - length (mat) returns either the number of rows or the number of columns in the matrix, whichever is larger.
  - size(mat) returns the number of rows and the number of columns of the matrix
  - size (vec) returns the number of rows and the number of columns of the vector.

Matrix Operations

- isequal (A, B): two matrices are considered equal if they have the same dimensions, and all corresponding elements equal.
- A+B, A-B : matris toplama ve çıkarma performed by adding or subtracting the corresponding elements. Two matrices must be the same size.
- A * c: Scalar matrix multiplication is performed by multiplying each element by the same scalar.
- diag (A): produces a column vector containing the elements of the main diagonal of A

```
>> A = magic (3)
   A =
        8   1   6
        3   5   7
        4   9   2
>> v = diag(A)
v =
     8
     5
     2
```

• diag(v) : produces a square diagonal matrix with the elements of vector v on the main diagonal.

```
>> B = diag(v)
   B =
        8   0   0
        0   5   0
        0   0   2
```

olur.

Matrix Vector Multiplication

• If A is an m x n matrix, and x is an n x 1 vector, then the "product of Ax" is a m x 1 vector.

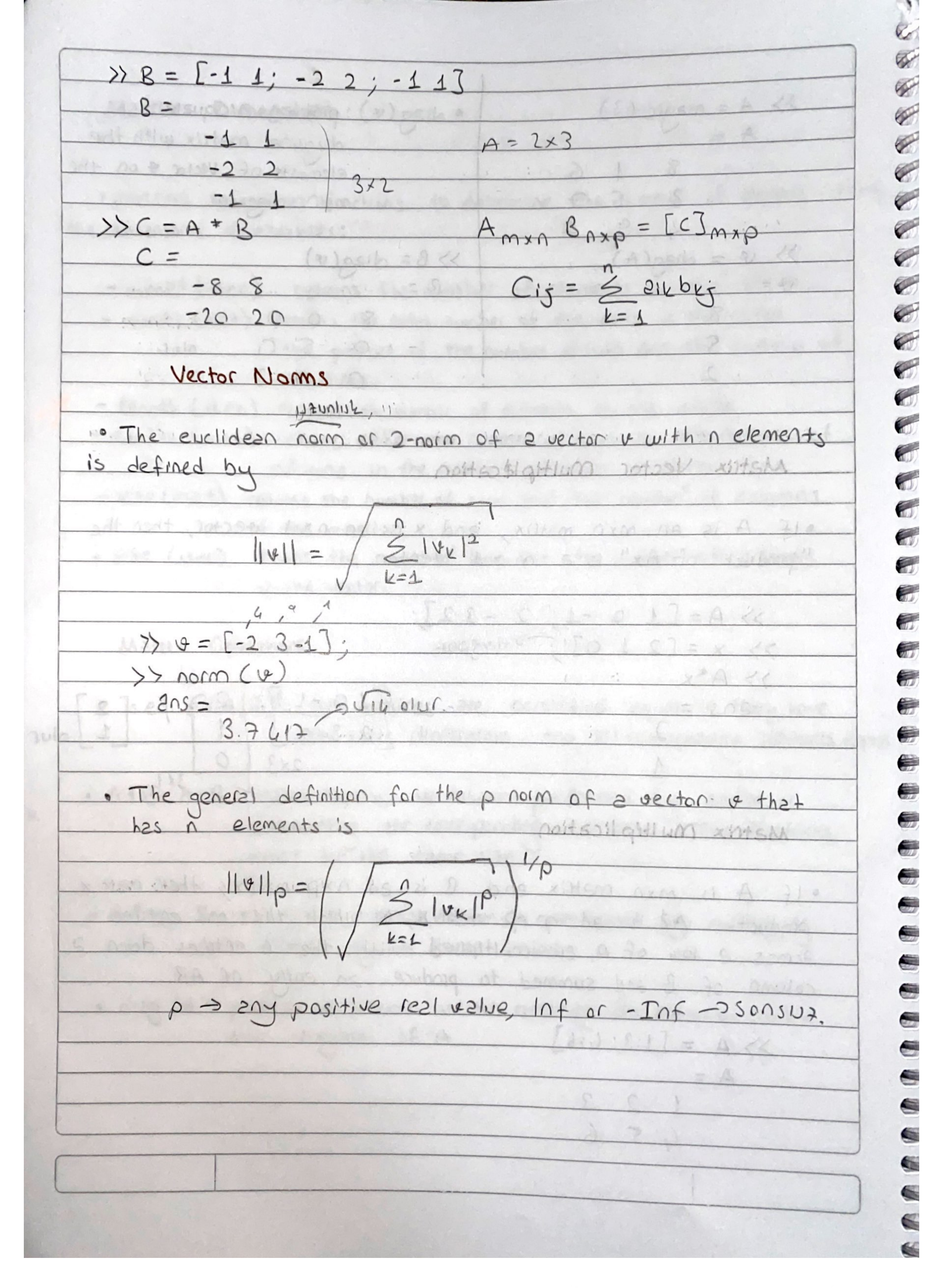
```
>> A = [1 0 -1; 2 -3 2];
>> x = [2 1 0]';   → transpoz
>> A*x
   ans =
         2
         1
```

$$\begin{bmatrix} 1 & 0 & -1 \\ 2 & -3 & 2 \end{bmatrix}_{2\times3} \begin{bmatrix} 2 \\ 1 \\ 0 \end{bmatrix}_{3\times1} \quad \begin{bmatrix} 2 \\ 1 \end{bmatrix}$$ olur

Matrix Multiplication

• If A is m x n matrix and B is an n x p matrix, their matrix product AB is an m x p matrix, in which the m entries across a row of A are multiplied with the n entries down a column of B and summed to produce an entry of AB.

```
>> A = [1:3; 4:6]
   A =
        1  2  3
        4  5  6
```



```
>> B = [-1 1; -2 2; -1 1]
B =
    -1   1
    -2   2
    -1   1
```

3x2

$A = 2\times3$

```
>> C = A * B
C =
    -8    8
   -20   20
```

$$A_{m\times n}\, B_{n\times p} = [C]_{m\times p}$$

$$C_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}$$

Vector Norms

- The euclidean norm (uzunluk) or 2-norm of a vector v with n elements is defined by

$$\|v\| = \sqrt{\sum_{k=1}^{n} |v_k|^2}$$

```
>> v = [-2 3 -1];
>> norm(v)
ans =
   3.7417
```

→ $\sqrt{14}$ olur.

- The general definition for the p norm of a vector v that has n elements is

$$\|v\|_p = \left(\sum_{k=1}^{n} |v_k|^p\right)^{1/p}$$

p → any positive real value, Inf or -Inf → sonsuz.

Determinant

mesela $A=\begin{pmatrix}1\\0\end{pmatrix}$ $B=\begin{pmatrix}2\\0\end{pmatrix}$ $C=\begin{pmatrix}2\\1\end{pmatrix}$ $D=\begin{pmatrix}1\\1\end{pmatrix}$

olsun

A, B, C, D yi $\begin{pmatrix}3 & 0\\1 & 1\end{pmatrix}$ ile her elemanını çarparsak aslında

0 ise alanı 0 olur. Doğru veya noktadır.

ABCD'nin alanı bunun determinantıyla çarpılırsa yeni alan olur.

Matris tersi, lineer işlemler vb. için kullanılabilir. $|A|$ ile gösterilir.

2x2 matris için;

$$A=\begin{bmatrix}a & b\\c & d\end{bmatrix} \qquad |A| = ad - bc$$

3x3 matris için;

$$A=\begin{bmatrix}a & b & c\\d & e & f\\g & h & i\end{bmatrix} \qquad |A| = a(ei-fh)+b(fg-di)+c(dh-eg)$$

şeklinde olur.

- Geometrik olarak yorumlarsak; determinant, area (2D), volume (3D) veya hypervolume (4D or higher) nın nasıl olduğunu X nokta kümesinin değişimine göre when matrix A is applied to X.

Sınavda matlab da yapılması muhtemel,
Bir problemden grafik çıkarma vb olabilir. Sorular özelleşecek.

Determinant bir şeklin alanının ne kadar değişeceğini gösterir. 3D şekil için hacminin, 4D hipervolume'nün ne kadar değişeceğini gösterir.

matlab'de det(matris) → şeklinde olur.

Matrix Inverse

Bir matrisin tersi, bu matris ile çarpıldığında birim matrisi elde ederiz.

$$A \cdot A^{-1} = I$$

```
>> A = [1 3; 2 4]
A =

   1   3
   2   4
```

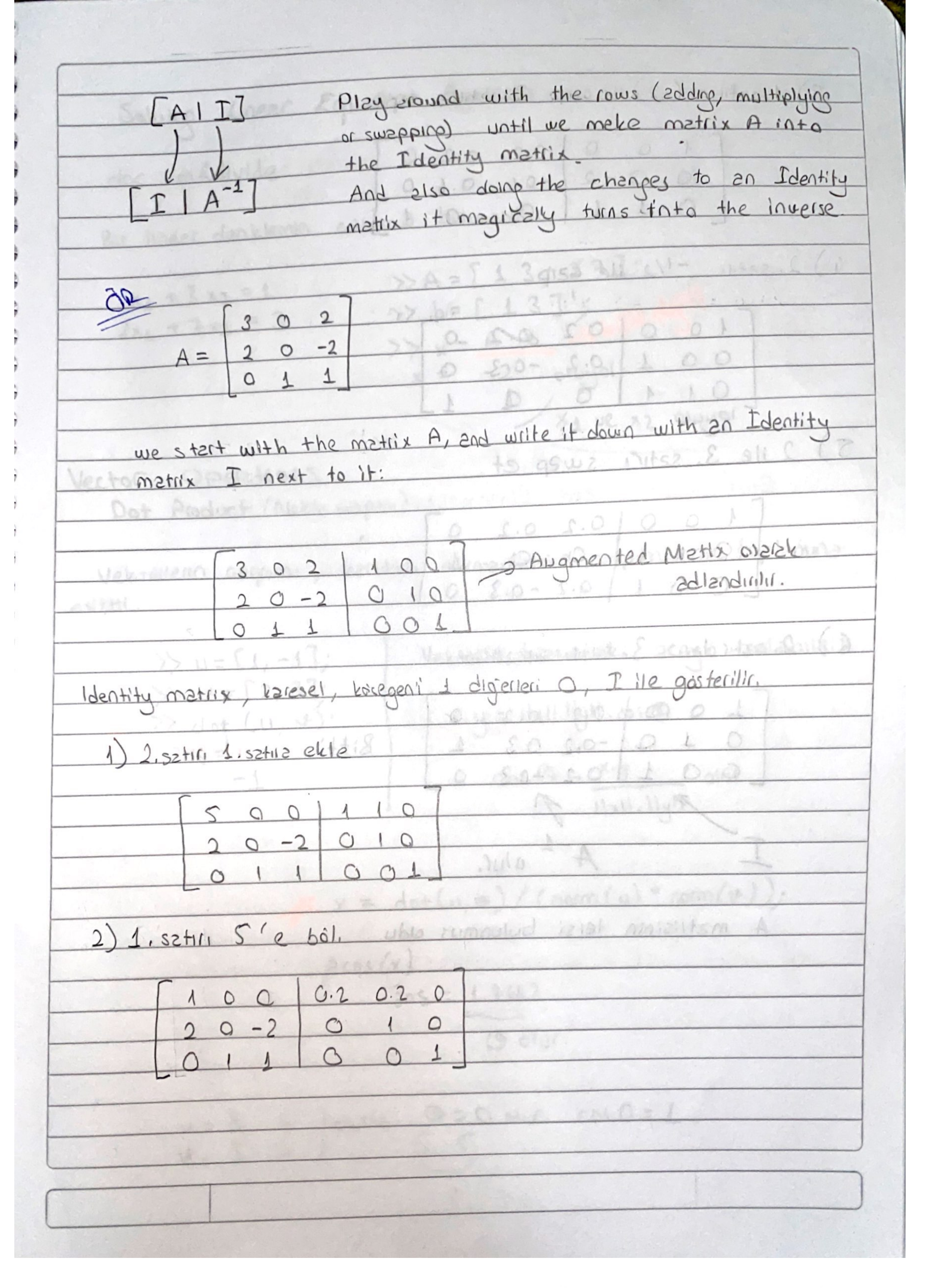
```
>> inv(A)
ans =
   -2.0000   1.5000
    1.0000  -0.5000
```

```
>> A * inv(A)
ans =
   1   0
   0   1
```

Extra NOT

mathisfun.com → güzel site.

Inverse of a Matrix using elementary Row operations.
(Also called Gauss-Jordan method)



$[A \mid I] \rightarrow [I \mid A^{-1}]$

Play around with the rows (adding, multiplying or swapping) until we make matrix A into the Identity matrix.
And also doing the changes to an Identity matrix it magically turns into the inverse.

Ör

$$A = \begin{bmatrix} 3 & 0 & 2 \\ 2 & 0 & -2 \\ 0 & 1 & 1 \end{bmatrix}$$

we start with the matrix A, and write it down with an Identity matrix I next to it:

$$\left[\begin{array}{ccc|ccc} 3 & 0 & 2 & 1 & 0 & 0 \\ 2 & 0 & -2 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \end{array}\right]$$

→ Augmented Matrix olarak adlandırılır.

Identity matrix, karesel, köşegeni 1 diğerleri 0, I ile gösterilir.

1) 2. satırı 1. satıra ekle

$$\left[\begin{array}{ccc|ccc} 5 & 0 & 0 & 1 & 1 & 0 \\ 2 & 0 & -2 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \end{array}\right]$$

2) 1. satırı 5'e böl.

$$\left[\begin{array}{ccc|ccc} 1 & 0 & 0 & 0.2 & 0.2 & 0 \\ 2 & 0 & -2 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \end{array}\right]$$

Solving Linear Equation Systems

doc mldivide

Bir lineer denklemin çözümü bulunabilir.

$x_1 + 3x_2 = 1$
$2x_1 + 7x_2 = 3$

```
>> A = [1 3; 2 7];
>> b = [1 3]';
>> x = A\b
x =   -2
       1
```

→ bölü değil

-2 oluyor, 1 → x_1 ve x_2 oluyor.

Vector Operations

Dot Product (Nokta çarpanı)

Vektörlerin çarpımı, karşılıklı elemanlarının çarpımlarının toplamına eşittir.

```
>> u=[1,-1];
>> v=[2,3];
>> dot(u,v);
ans =
      -1
```

Vektörler arasındaki açıyı bulabiliriz.

$x.y = ||x||.||y||.\cos\theta$

$\cos\theta = \frac{x.y}{||x||.||y||}$ olur.

```
x = dot(u,v) / (norm(u) * norm(v));
```

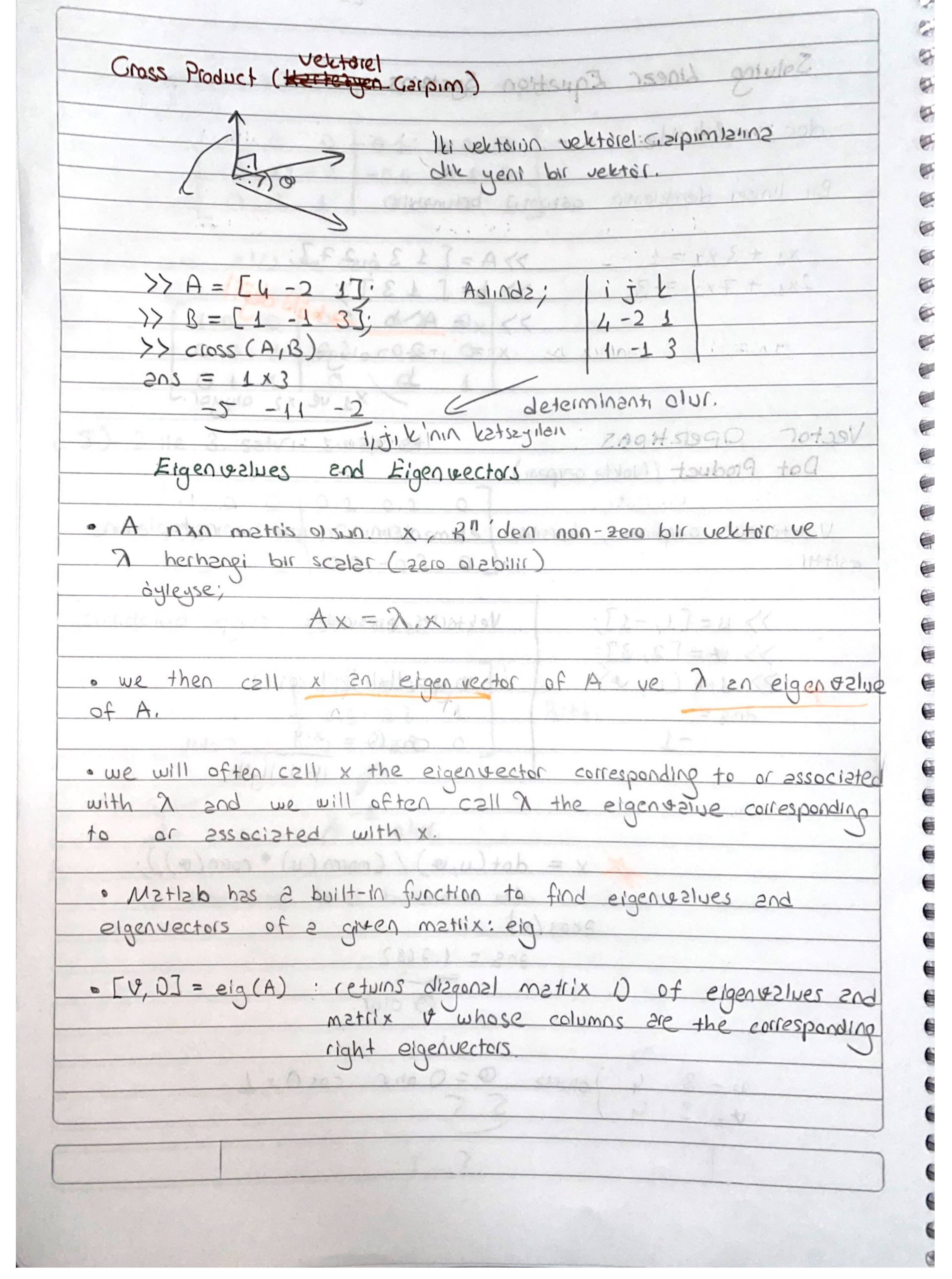
```
acos(x)
    ans = 1.7682
```

θ olur.

u = 3 4, v = 3 4 olursa θ = 0 olur. cos 0 = 1

5 . 5

5 . 5



Cross Product (vektörel ~~kartezyen~~ çarpım)

İki vektörün vektörel çarpımlarına dik yeni bir vektör.

```
>> A = [4 -2 1];
>> B = [1 -1 3];
>> cross(A,B)
ans = 1x3
   -5  -11  -2
```

i, j, k'nın katsayıları

Aslında;

$$\begin{vmatrix} i & j & k \\ 4 & -2 & 1 \\ 1 & -1 & 3 \end{vmatrix}$$

determinantı olur.

Eigenvalues and Eigenvectors

- A nxn matris olsun. x, R^n'den non-zero bir vektör ve λ herhangi bir scalar (zero olabilir) öyleyse;

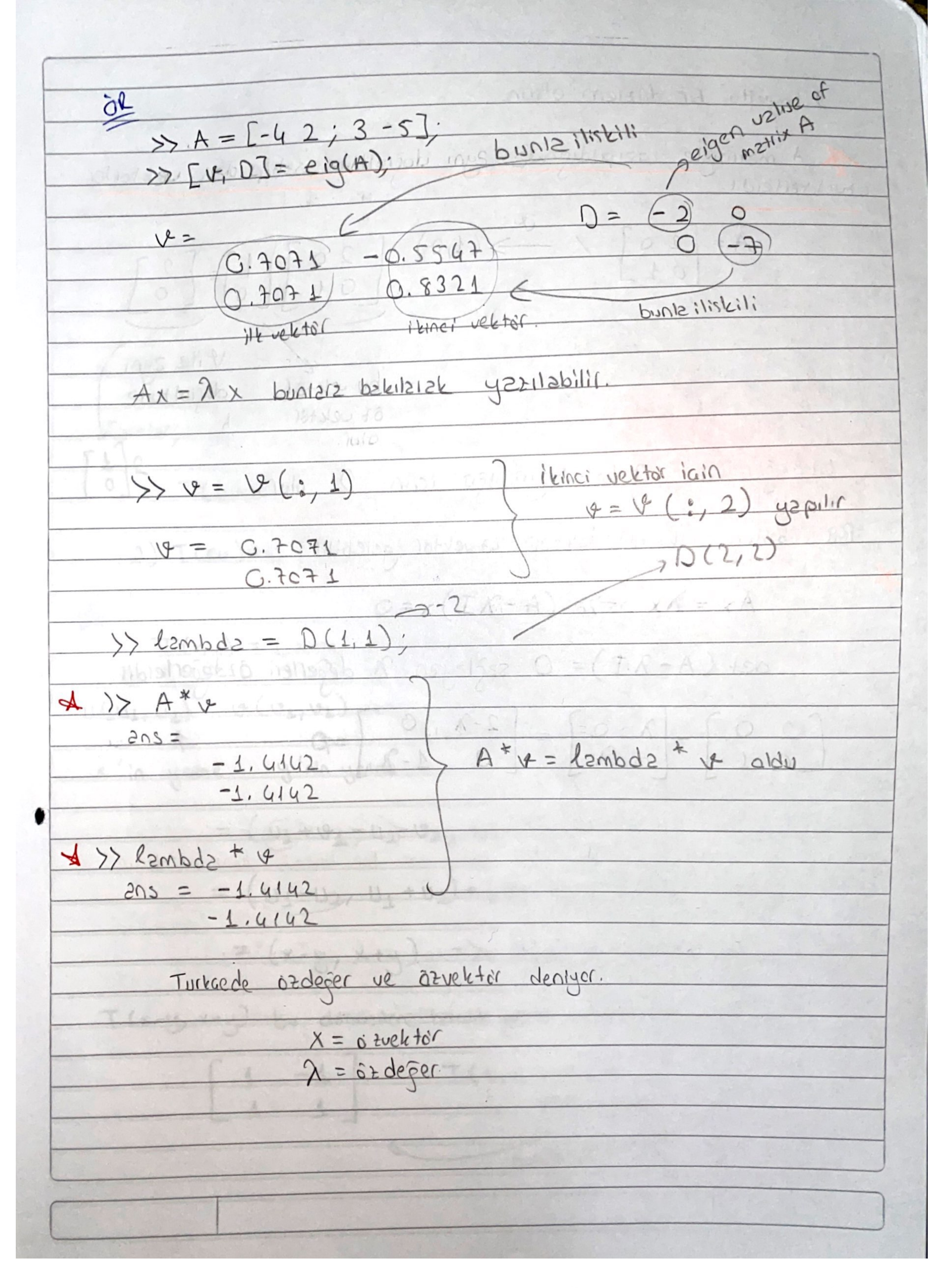
$$Ax = \lambda . x$$

- we then call x an eigen vector of A ve λ an eigen value of A.

- we will often call x the eigenvector corresponding to or associated with λ and we will often call λ the eigenvalue corresponding to or associated with x.

- Matlab has a built-in function to find eigenvalues and eigenvectors of a given matrix: eig

- [V, D] = eig(A) : returns diagonal matrix D of eigenvalues and matrix V whose columns are the corresponding right eigenvectors.



Ör

```
>> A = [-4 2 ; 3 -5];
>> [v, D] = eig(A);
```

bunla ilişkili

eigen value of matrix A

v =

0.7071 -0.5547
0.7071 0.8321

ilk vektör — ikinci vektör

D = -2 0
0 -7

bunla ilişkili

$Ax = \lambda x$ bunlara bakılarak yazılabilir.

```
>> v = v(:, 1)

v = 0.7071
    0.7071
```

ikinci vektör için
v = v(:, 2) yapılır

-2

```
>> lambda = D(1,1);
```

D(2,2)

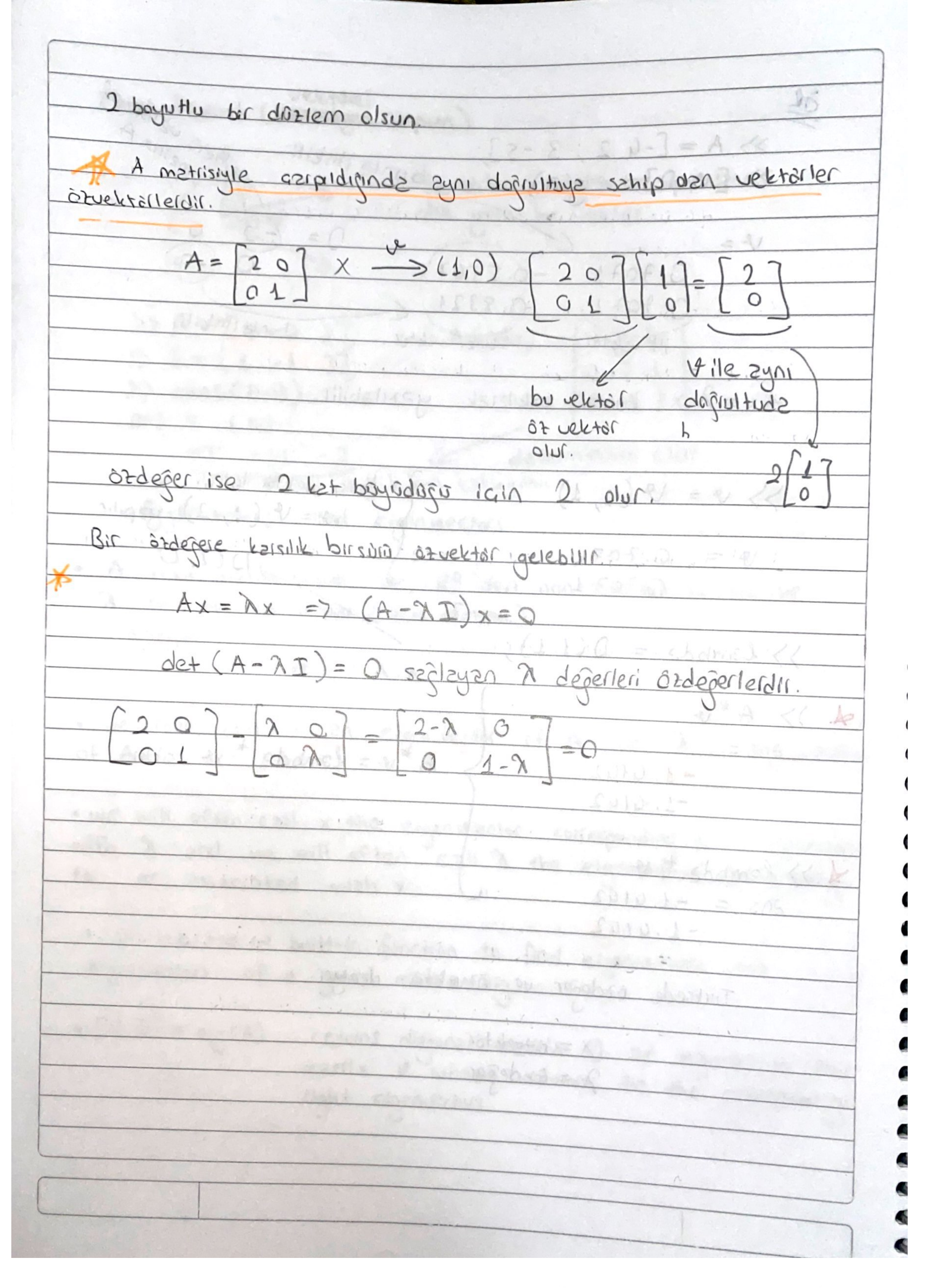
```
>> A*v
ans =
    -1.4142
    -1.4142

>> lambda * v
ans = -1.4142
      -1.4142
```

A*v = lambda * v oldu

Türkçede özdeğer ve özvektör deniyor.

x = özvektör
λ = özdeğer



2 boyutlu bir düzlem olsun.

A matrisiyle çarpıldığında aynı doğrultuya sahip olan vektörler özvektörlerdir.

$A = \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix}$ x $\xrightarrow{v} (1,0)$ $\quad \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 2 \\ 0 \end{bmatrix}$

bu vektör öz vektör olur.

v ile aynı doğrultuda → $2\begin{bmatrix} 1 \\ 0 \end{bmatrix}$

özdeğer ise 2 kat büyüdüğü için 2 olur.

Bir özdeğere karşılık birsürü özvektör gelebilir.

$$Ax = \lambda x \Rightarrow (A - \lambda I)x = 0$$

$\det(A - \lambda I) = 0$ sağlayan λ değerleri özdeğerlerdir.

$$\begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix} - \begin{bmatrix} \lambda & 0 \\ 0 & \lambda \end{bmatrix} = \begin{bmatrix} 2-\lambda & 0 \\ 0 & 1-\lambda \end{bmatrix} = 0$$

Lineer Dönüşümler (Lineer Transformations)

n boyutlu reel sayılar uzayında m boyutlu reel sayılar uzayına bir vektör vb. dönüştüren onu izdüşen yeni bir vektör bulan dönüşümlerdir. Lineer olması için şartlar var.

$T(x,y) = (x^2, x-y, y^2)$

$T(1,2) = (1,-1,4)$ × Bu lineer dönüşüm mü.

Şartlar;

1) $T(u+v) = T(u) + T(v)$ for all u, v; in the domain of T.

2) $T(cu) = c\,T(u)$ for all scalars c and all u in the domain of T.

ÖR

$T(x,y) = (x-y, x+y)$ olsun.

$u(u_1, u_2)$ $v(v_1, v_2)$ ise;

x'in yerine — y'nin yerine

$= (u_1 + v_1 - u_2 - v_2,\ u_1 + v_1 + u_2 + v_2)$

$= (u_1 - u_2,\ u_1 + u_2) + (v_1 - v_2,\ v_1 + v_2)$

$= (x-y, x+y) \Rightarrow T(u) + T(v)$ oldu sağladı.

$T(x-y, x+y)$ bu dönüşüme karşılık gelen matris; katsayılarıdır.

$$\begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix}$$

$T(1,4) = (3,5)$ olur yada

bunla çarpılabilir.

$$\begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 4 \end{bmatrix} = \begin{bmatrix} -3 \\ 5 \end{bmatrix}$$

extra not;

Lineer Dönüşüm

Lineer dönüşümler, mühendislik ve fizik sahalarında sıkça kullanılan ve matrisler ile ifade edilebilmesinden dolayı kullanımı oldukça pratik fonksiyonlardır. Özellikle hareket ifade eden lineer dönüşümler; bilgisayar grafikleri, makine hareketleri, animasyon ve robot teknolojisinde sıkça kullanılmaktadır.

Lineer dönüşümler, $\mathbb{R}^n$ uzayındaki elemanları, noktaları, vektörleri $\mathbb{R}^m$ uzayındaki elemanlara, noktalara, vektörlere dönüştüren özel fonksiyonlardır:

$$f: \mathbb{R}^n \rightarrow \mathbb{R}^m$$

Tanım: $T: \mathbb{R}^n \rightarrow \mathbb{R}^m$ dönüşümü, her $\vec{u}, \vec{v} \in \mathbb{R}^n$ ve $c \in \mathbb{R}$ için

$$T(\vec{u} + \vec{v}) = T(\vec{u}) + T(\vec{v})$$

$$T(c\vec{u}) = cT(\vec{u})$$

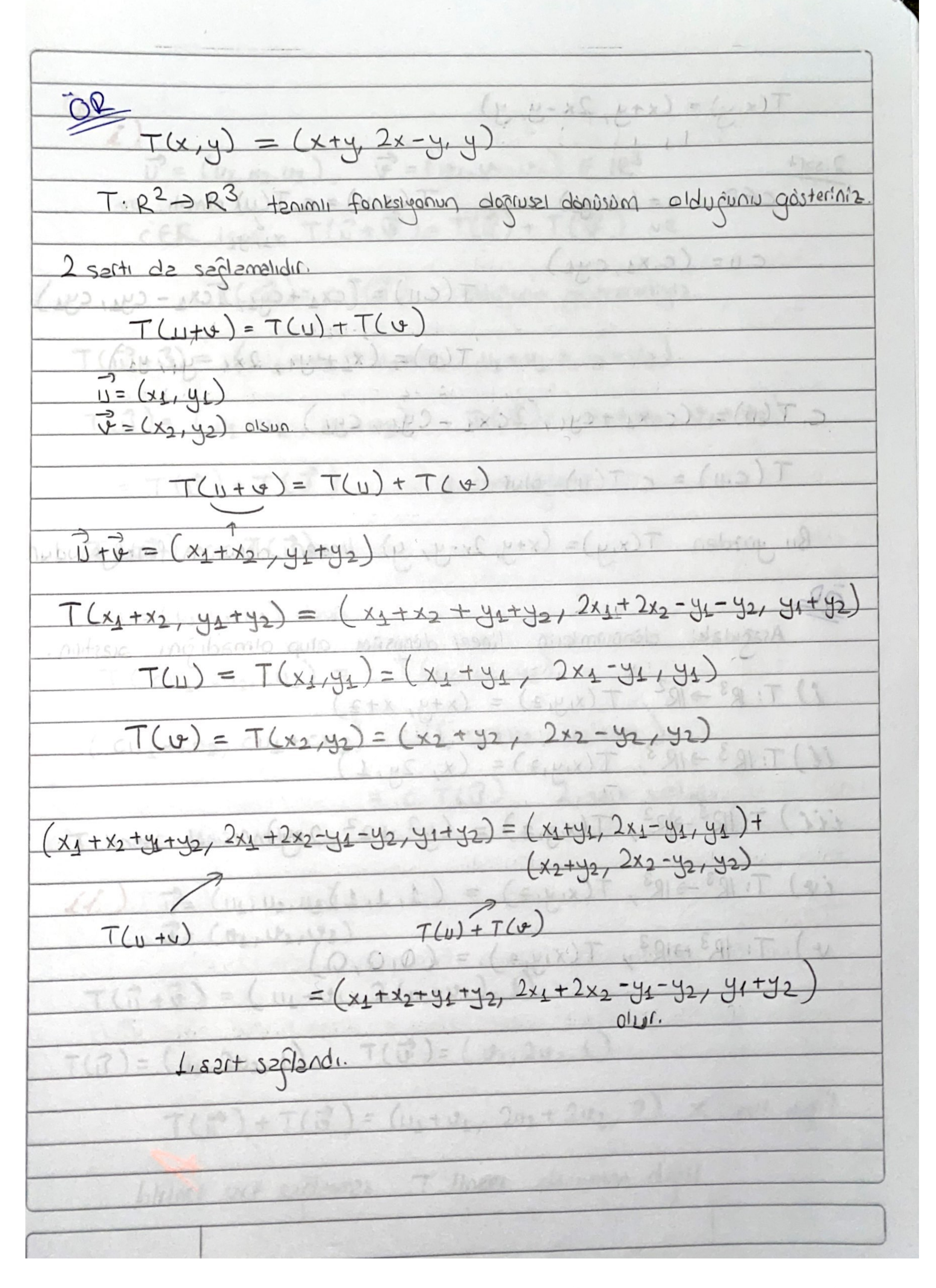
koşullarını sağlıyorsa $\mathbb{R}^n$'den $\mathbb{R}^m$'ye bir lineer dönüşüm denir.

Ör

→ tek boyutlu

$$T: \mathbb{R}^3 \rightarrow \mathbb{R} \rightarrow T(x_1, x_2, x_3) = (x_1 * x_2 * x_3)$$

$$T: \mathbb{R}^2 \rightarrow \mathbb{R}^3 \rightarrow T(x_1, x_2) = (x_1 + x_2, x_1 - x_2, x_1 . x_2)$$



ÖR

$$T(x,y) = (x+y,\ 2x-y,\ y)$$

$T: R^2 \to R^3$ tanımlı fonksiyonun doğrusal dönüşüm olduğunu gösteriniz.

2 şartı da sağlamalıdır.

$$T(u+v) = T(u) + T(v)$$

$\vec{u} = (x_1, y_1)$
$\vec{v} = (x_2, y_2)$ olsun.

$$T(u+v) = T(u) + T(v)$$

$$\vec{u} + \vec{v} = (x_1 + x_2,\ y_1 + y_2)$$

$$T(x_1+x_2,\ y_1+y_2) = (x_1+x_2+y_1+y_2,\ 2x_1+2x_2-y_1-y_2,\ y_1+y_2)$$

$$T(u) = T(x_1,y_1) = (x_1+y_1,\ 2x_1-y_1,\ y_1)$$

$$T(v) = T(x_2,y_2) = (x_2+y_2,\ 2x_2-y_2,\ y_2)$$

$$(x_1+x_2+y_1+y_2,\ 2x_1+2x_2-y_1-y_2,\ y_1+y_2) = (x_1+y_1,\ 2x_1-y_1,\ y_1) + (x_2+y_2,\ 2x_2-y_2,\ y_2)$$

$T(u+v)$ $\qquad$ $T(u)+T(v)$

$$= (x_1+x_2+y_1+y_2,\ 2x_1+2x_2-y_1-y_2,\ y_1+y_2)$$ olur.

1. şart sağlandı.

$T(x,y) = (x+y,\ 2x-y,\ y)$

2. şart

$c \in \mathbb{R}$ $\quad u = (x_1, y_1)$ olsun $\quad T(c.u) = c.T(u)$ olmalı.

$\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad x_1, y_1$

$c.u = (c.x_1, cy_1)$

$T(c.u) = (cx_1 + cy_1,\ 2cx_1 - cy_1,\ cy_1)$

$T(u) = (x_1 + y_1,\ 2x_1 - y_1,\ y_1)$

$c.T(u) = (c.x_1 + cy_1,\ 2cx_1 - cy_1,\ cy_1)$

$T(c.u) = c.T(u)$ olur.

Bu yüzden $T(x,y) = (x+y,\ 2x-y,\ y)$ lineer dönüşüm fonksiyonudur.

ÖR

Aşağıdaki dönüşümlerin lineer dönüşüm olup olmadığını araştırın.

i) $T: \mathbb{R}^3 \to \mathbb{R}^2$, $T(x,y,z) = (x+y,\ x+z)$

ii) $T: \mathbb{R}^3 \to \mathbb{R}^3$, $T(x,y,z) = (x,\ 2y,\ 1)$

iii) $T: \mathbb{R}^3 \to \mathbb{R}^2$, $T(x,y,z) = (y^2 + z^2,\ 2xy)$

iv) $T: \mathbb{R}^3 \to \mathbb{R}^3$, $T(x,y,z) = (1,1,1)$

v) $T: \mathbb{R}^3 \to \mathbb{R}^3$, $T(x,y,z) = (0,0,0)$

Çöz

i) $\vec{u} = (u_1, u_2, u_3)$, $\vec{v} = (v_1, v_2, v_3) \in \mathbb{R}^3$

ve $c \in \mathbb{R}$ için $T(\vec{u}+\vec{v}) = T(\vec{u}) + T(\vec{v})$ ve

$T(c\vec{u}) = c\,T(\vec{u})$ olduğunu göstermeliyiz.

$T(\vec{u}+\vec{v}) = (u_1+v_1+u_2+v_2,\ u_1+v_1+u_3+v_3)$

$T(\vec{u}) = (u_1+u_2,\ u_1+u_3)$ $\quad T(\vec{v}) = (v_1+v_2,\ v_1+v_3)$

$= T(\vec{u}) + T(\vec{v})$ $\quad$ 1. şart sağlanır.

$T(c\vec{u}) = c.T(\vec{u})$

$u = (u_1, u_2, u_3)$ olsun.

$T(cu_1, cu_2, cu_3) = c.T(u_1, u_2, u_3)$

$(cu_1+cu_2,\ cu_1+cu_3) = c.(u_1+u_2,\ u_1+u_3)$

$= c.T(\vec{u})$ $\quad$ 2. şart sağlanır.

T lineer dönüşümdür.

ii) $\vec{u} = (u_1, u_2, u_3)$
$\vec{v} = (v_1, v_2, v_3)$

$T(\vec{u}+\vec{v}) = (u_1+v_1,\ 2(u_2+v_2),\ 1)$

$T(\vec{u}) = (u_1, 2u_2, 1)$ $\quad T(\vec{v}) = (v_1, 2v_2, 1)$

$T(\vec{u}) + T(\vec{v}) = (u_1+v_1,\ 2u_2+2v_2,\ 2)$ × eşit değil

birinci şart sağlanmaz. T lineer dönüşüm değil.

iii)

$\vec{u} = (u_1, u_2, u_3) \quad \vec{v} = (v_1, v_2, v_3)$

$T(\vec{u}+\vec{v}) = ((u_2+v_2)^2 + (u_3+v_3)^2, \ 2(u_1+v_1)(u_2+v_2))$ olur

hoca buna bakmamış.

$T(c\vec{u}) = (c^2u_2^2 + c^2u_3^2, \ 2c^2 u_1 . u_2)$ → Bu $c^2 . T(\vec{u})$ olur.

$c . T(\vec{u}) = c(u_2^2 + u_3^2, \ 2u_1 . u_2)$

eşit olmaz

birinci şartta da lineer olmadığı görülür.

iv)

$T(\vec{u}+\vec{v}) = T(u_1+v_1, u_2+v_2, u_3+v_3) = (1,1,1)$

$T(\vec{u}) = T(u_1, u_2, u_3) = (1,1,1)$ Ne koyarsak sabit olacak

$T(\vec{v}) = T(v_1, v_2, v_3) = (1,1,1)$

$(2,2,2)$

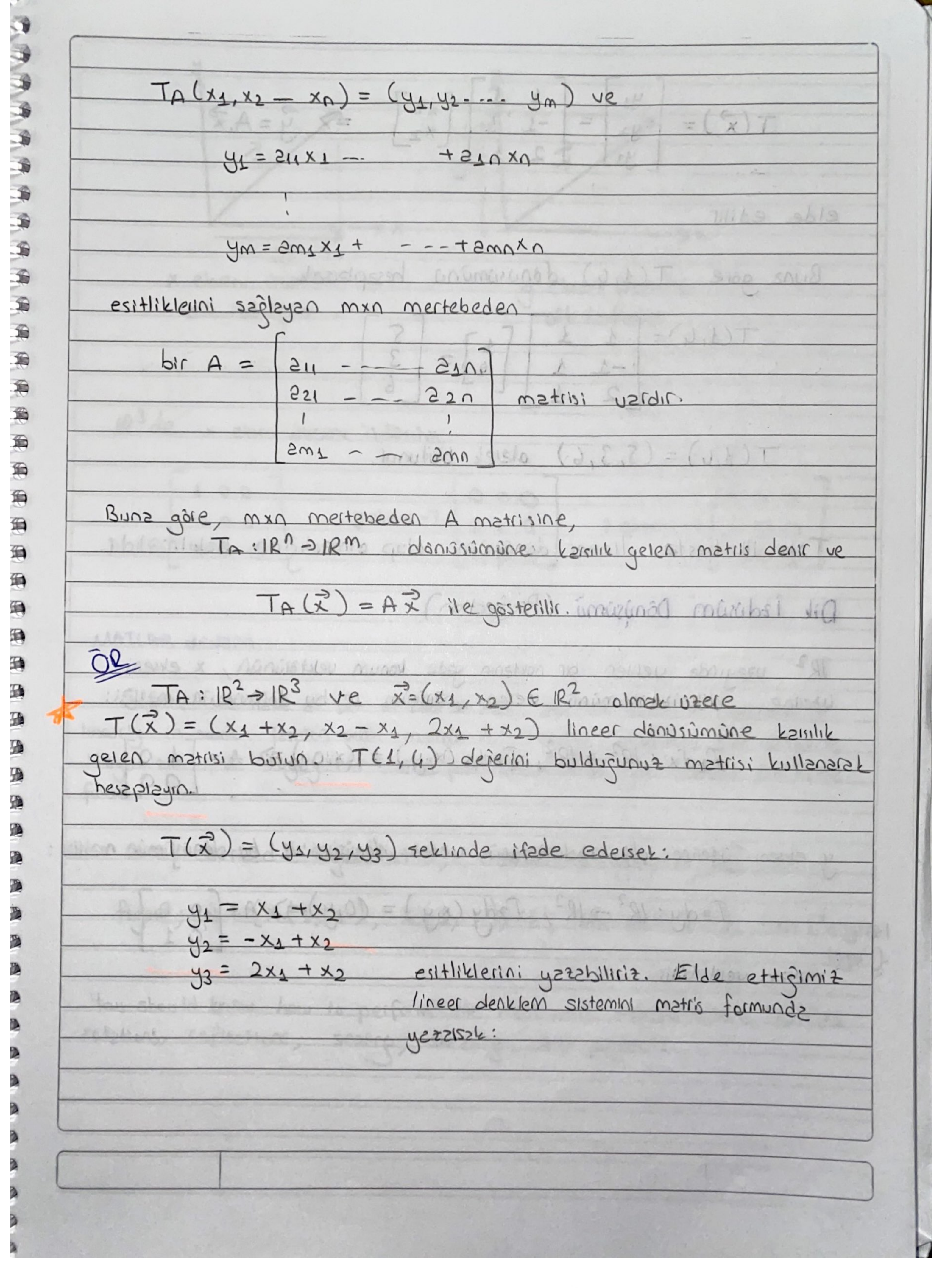
Lineer dönüşüm değildir.

v) $T: \mathbb{R}^3 \to \mathbb{R}^3, \ T(x,y,z) = (0,0,0)$ dönüşümü 0 dönüşümüdür. $\mathbb{R}^n$'de tanımlı her $\vec{x}$ vektörünü $\mathbb{R}^m$'de 0 vektörü ile eşler ve lineer bir dönüşümdür. Lineer

Teorem

Lineer Dönüşümün Matris Gösterimi

$T_A: \mathbb{R}^n \to \mathbb{R}^m$ bir lineer dönüşüm olmak üzere, her $\vec{x} = (x_1, ... x_n)$ vektörü için



$T_A(x_1, x_2 - x_n) = (y_1, y_2 - \cdots y_m)$ ve

$y_1 = a_{11}x_1 - \quad + a_{1n}x_n$

$\vdots$

$y_m = a_{m1}x_1 + \quad - - - + a_{mn}x_n$

eşitliklerini sağlayan $m \times n$ mertebeden

bir $A = \begin{bmatrix} a_{11} & - - - & a_{1n} \\ a_{21} & - - - & a_{2n} \\ \vdots & & \vdots \\ a_{m1} & - - - & a_{mn} \end{bmatrix}$ matrisi vardır.

Buna göre, $m \times n$ mertebeden A matrisine, $T_A : \mathbb{R}^n \to \mathbb{R}^m$ dönüşümüne karşılık gelen matris denir ve

$T_A(\vec{x}) = A\vec{x}$ ile gösterilir.

ÖR

$T_A : \mathbb{R}^2 \to \mathbb{R}^3$ ve $\vec{x} = (x_1, x_2) \in \mathbb{R}^2$ olmak üzere $T(\vec{x}) = (x_1 + x_2, x_2 - x_1, 2x_1 + x_2)$ lineer dönüşümüne karşılık gelen matrisi bulun. $T(1,4)$ değerini bulduğunuz matrisi kullanarak hesaplayın.

$T(\vec{x}) = (y_1, y_2, y_3)$ şeklinde ifade edersek:

$y_1 = x_1 + x_2$
$y_2 = -x_1 + x_2$
$y_3 = 2x_1 + x_2$ eşitliklerini yazabiliriz. Elde ettiğimiz lineer denklem sistemini matris formunda yazarsak:

$$T(\vec{x}) = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ -1 & 1 \\ 2 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \Rightarrow \vec{y} = A.\vec{x}$$

elde edilir.

Buna göre $T(1,4)$ dönüşümünü hesaplarsak;

$$T(1,4) = \begin{bmatrix} 1 & 1 \\ -1 & 1 \\ 2 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 4 \end{bmatrix} = \begin{bmatrix} 5 \\ 3 \\ 6 \end{bmatrix}$$

$T(1,4) = (5,3,6)$ olarak bulunur.

Tabi ilk başta lineer dönüşüm olup olmadığına bakılmalıdır.

Dik İzdüşüm Dönüşümü (Projection)

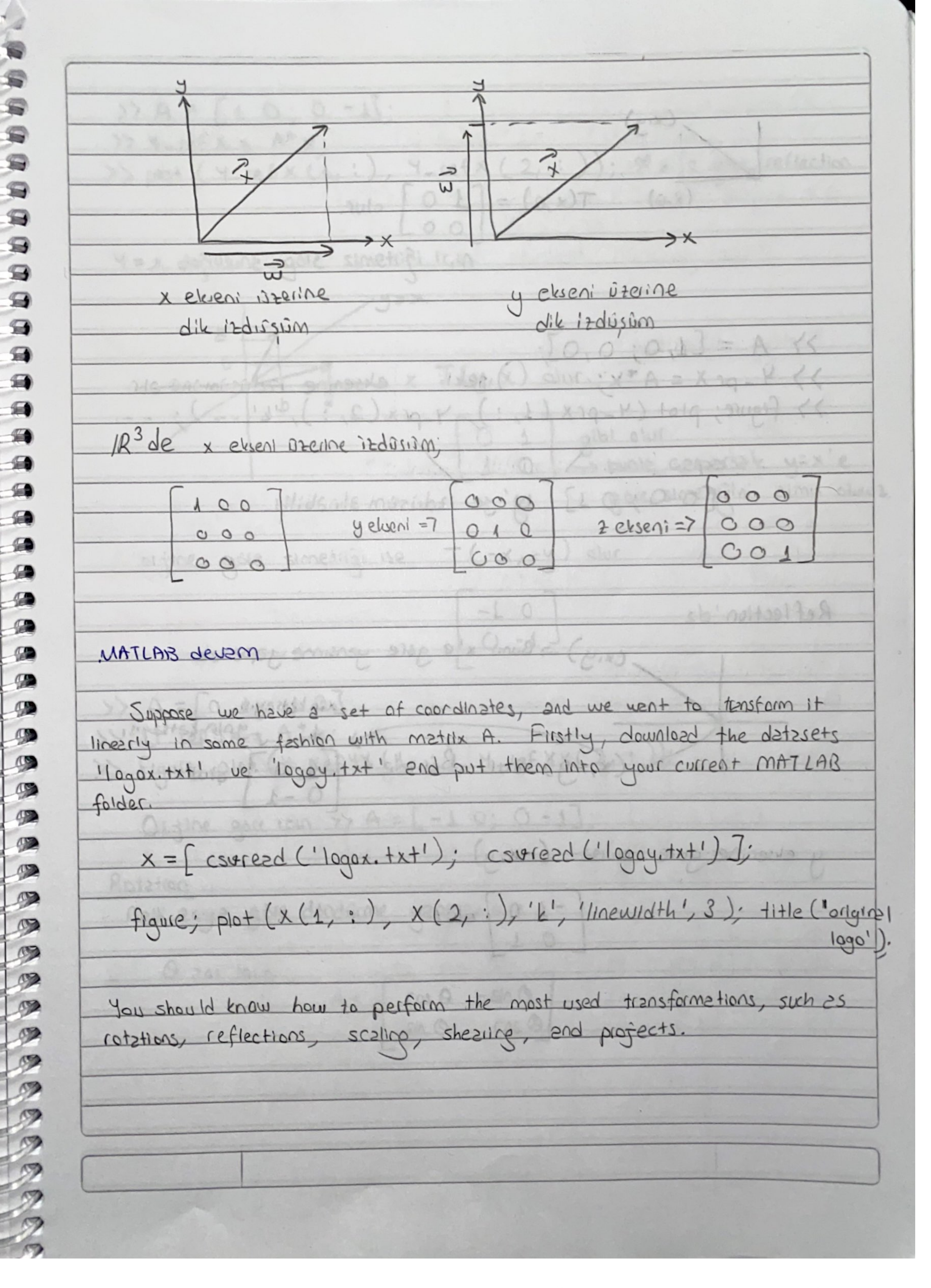
$\mathbb{R}^2$ uzayında verilen bir noktanın yada konum vektörünün, x ekseni üzerine dik izdüşümünü veren dönüşüm ve bu dönüşümün matrisi:

$$Izdx : \mathbb{R}^2 \to \mathbb{R}^2, \; Izdx(x,y) = (x,0) \Rightarrow A = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$$

y ekseni üzerine dik izdüşümünü veren dönüşüm ve bu dönüşümün matrisi

$$Izdy : \mathbb{R}^2 \to \mathbb{R}^2, \; Izdy(x,y) = (0,y) \Rightarrow A = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}$$

olarak verilebilir.



x ekseni üzerine dik izdüşüm

y ekseni üzerine dik izdüşüm

$\mathbb{R}^3$ de x ekseni üzerine izdüşüm;

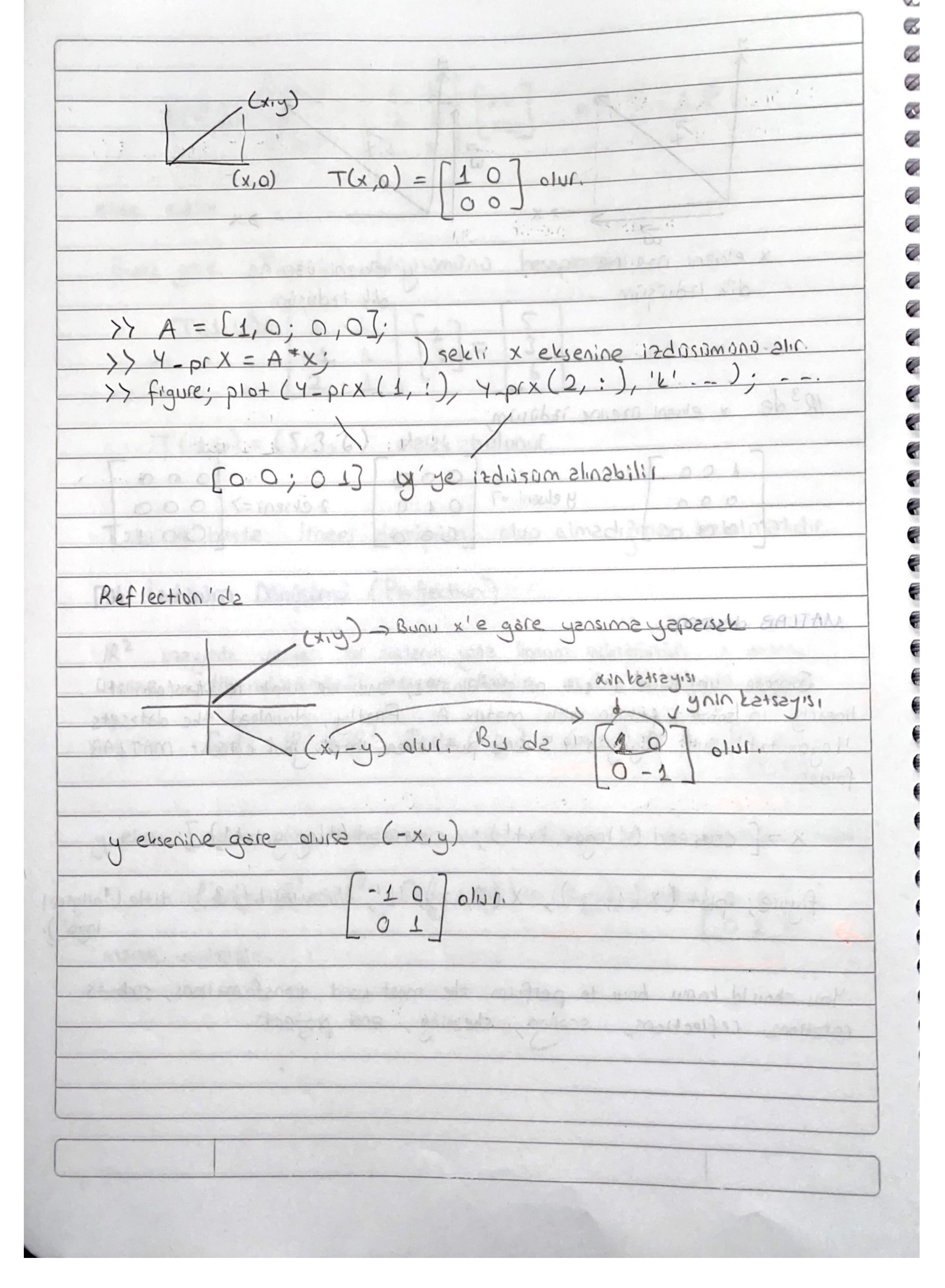
$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad y \text{ ekseni} \Rightarrow \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad z \text{ ekseni} \Rightarrow \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

MATLAB devam

Suppose we have a set of coordinates, and we want to transform it linearly in some fashion with matrix A. Firstly, download the datasets 'logox.txt' ve 'logoy.txt' and put them into your current MATLAB folder.

```
x = [ csvread('logox.txt'); csvread('logoy.txt') ];
figure; plot(x(1,:), x(2,:), 'k', 'linewidth', 3); title('original logo');
```

You should know how to perform the most used transformations, such as rotations, reflections, scaling, shearing, and projects.



(x,y)

$(x,0)$ $\quad T(x,0) = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$ olur.

```
>> A = [1,0; 0,0];
>> Y_prX = A*X;          ) şekli x eksenine izdüşümünü alır
>> figure; plot(Y_prX(1,:), Y_prX(2,:), 'k' ...); ...
```

[0 0; 0 1] y'ye izdüşüm alınabilir

Reflection'da

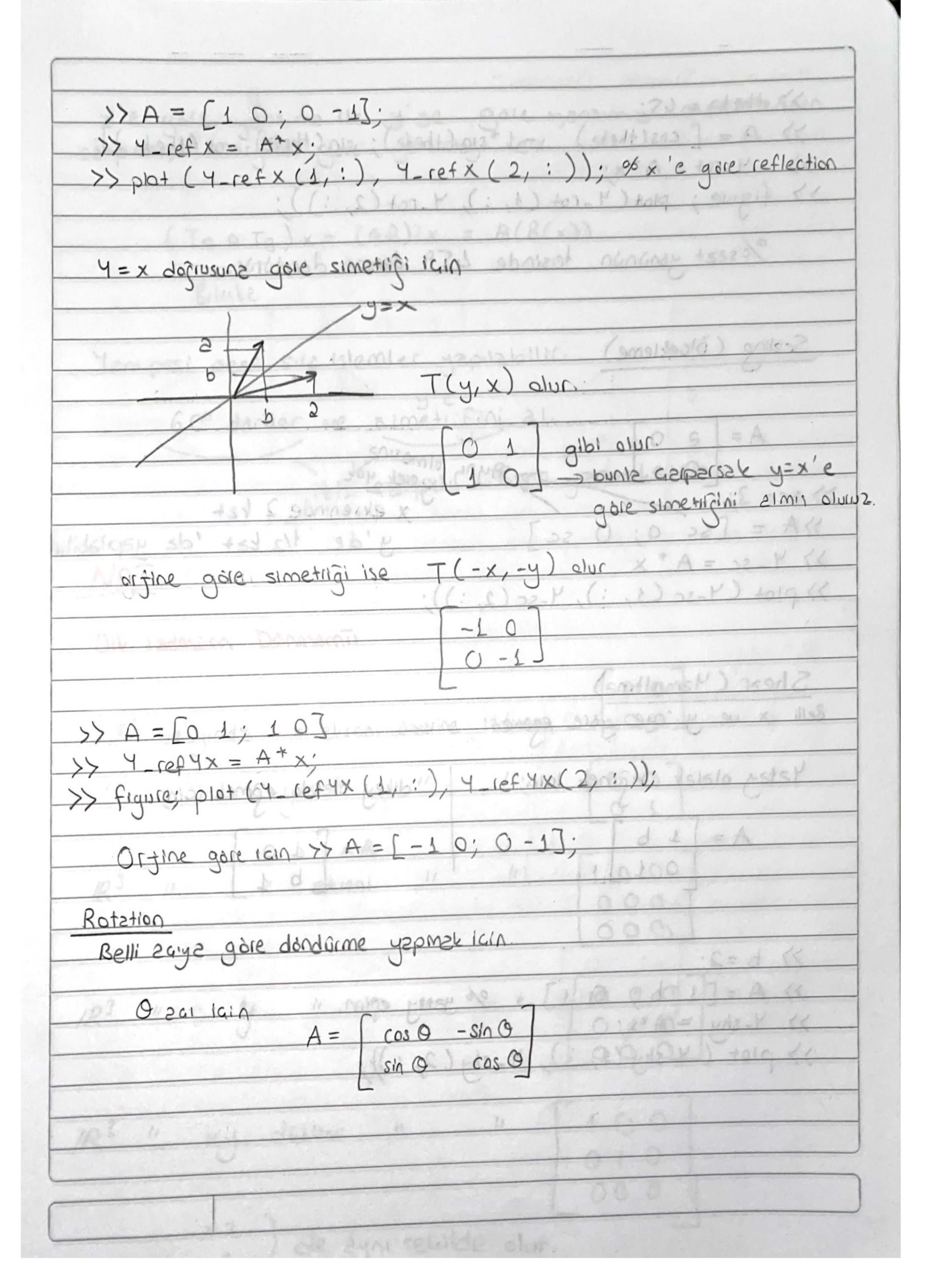
(x,y) → Bunu x'e göre yansıma yaparsak

$(x,-y)$ olur. Bu da $\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$ olur.

(x'in katsayısı, y'nin katsayısı)

y eksenine göre olursa $(-x,y)$

$\begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$ olur.



```
>> A = [1 0; 0 -1];
>> y_refx = A*x;
>> plot (y_refx(1,:), y_refx(2,:)); % x'e göre reflection
```

y = x doğrusuna göre simetriği için

T(y, x) olur.

$$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$ gibi olur.
→ bunla çarparsak y=x'e göre simetriğini almış oluruz.

orjine göre simetriği ise T(-x, -y) olur

$$\begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}$$

```
>> A = [0 1; 1 0]
>> y_refyx = A*x;
>> figure; plot (y_refyx(1,:), y_refyx(2,:));
```

Orjine göre için >> A = [-1 0; 0 -1];

Rotation

Belli açıya göre döndürme yapmak için.

θ açı için

$$A = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

```
>> theta = 45;
>> A = [cos(theta)   -1*sin(theta); sin(theta)  cos(theta)];
>> y_rot = A*x;
>> figure; plot(y_rot(1,:), y_rot(2,:));
```

% saat yönünün tersinde 45° derece döndürür.

Scaling (ölçekleme)

$$A = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$$

Aynı olmasına gerek yok
x ekseninde 2 kat
y'de 1/2 kat 'da yapılabilir

```
>> sc= 3;
>> A = [sc  0;  0  sc]
>> y_sc = A * x
>> plot (y_sc(1, :), y_sc(2, :));
```

Shear (Yamultma)

Belli x ve y'ye göre eğme

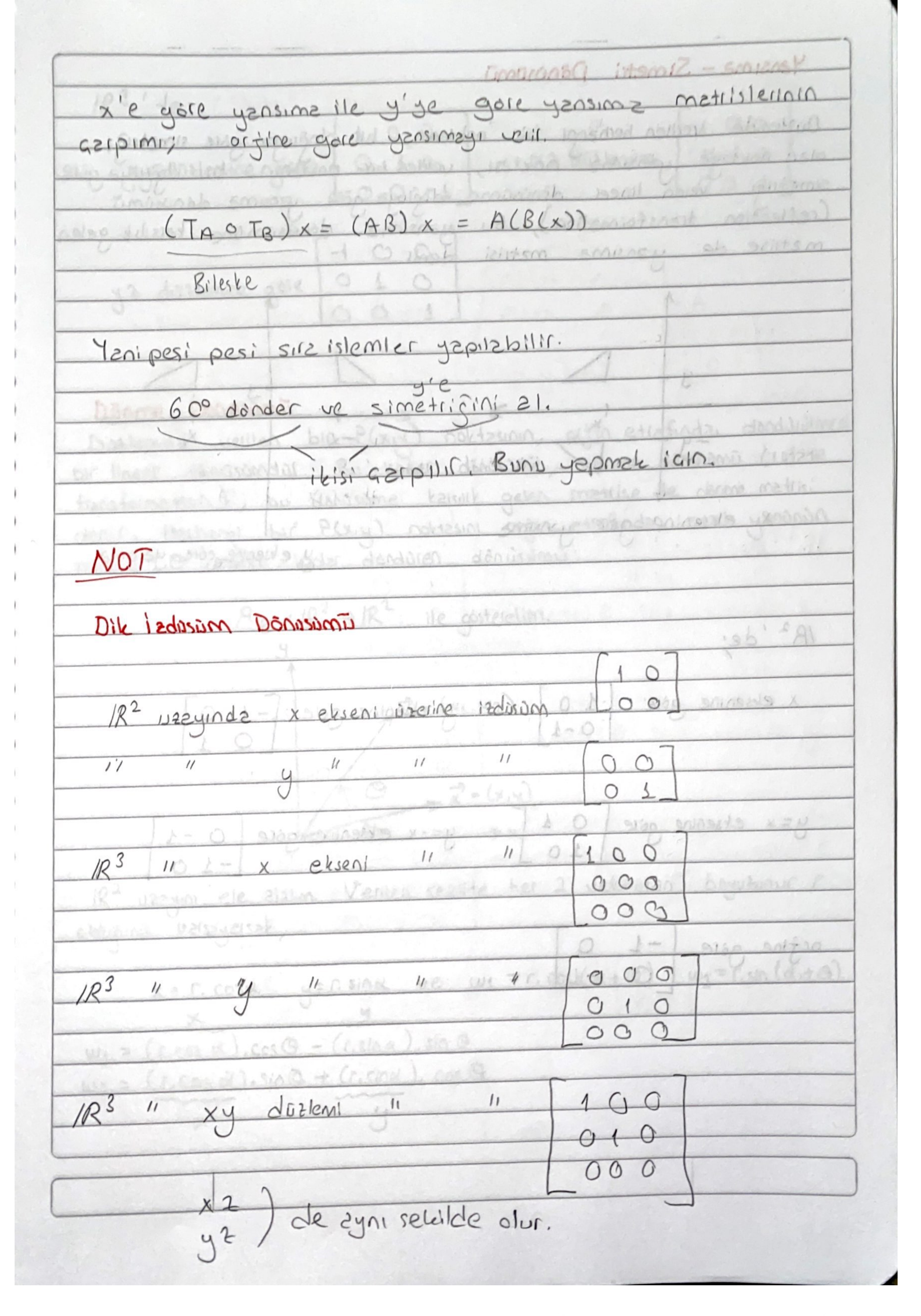
Yatay olarak eğmek için;

$$A = \begin{bmatrix} 1 & b \\ 0 & 1 \end{bmatrix}$$

dikey olarak eğmek için;

$$A = \begin{bmatrix} 1 & 0 \\ b & 1 \end{bmatrix}$$

```
>> b =2;
>> A =[1  b;  0  1]     % yatay eğim.
>> y_shy  = A*x;
>> plot ( y_shy (1, :), y_shy (2, :));
```



x'e göre yansıma ile y'ye göre yansıma matrislerinin çarpımı; orjine göre yansımayı verir.

$(T_A \circ T_B)x = (AB)x = A(B(x))$

Bileşke

Yani peşi peşi sıra işlemler yapılabilir.

60° döndür ve y'ye simetriğini al.

ikisi çarpılır. Bunu yapmak için.

NOT

Dik İzdüşüm Dönüşümü

$\mathbb{R}^2$ uzayında x ekseni üzerine izdüşüm $\begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$

" " y " " " $\begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}$

$\mathbb{R}^3$ " x ekseni " " $\begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$

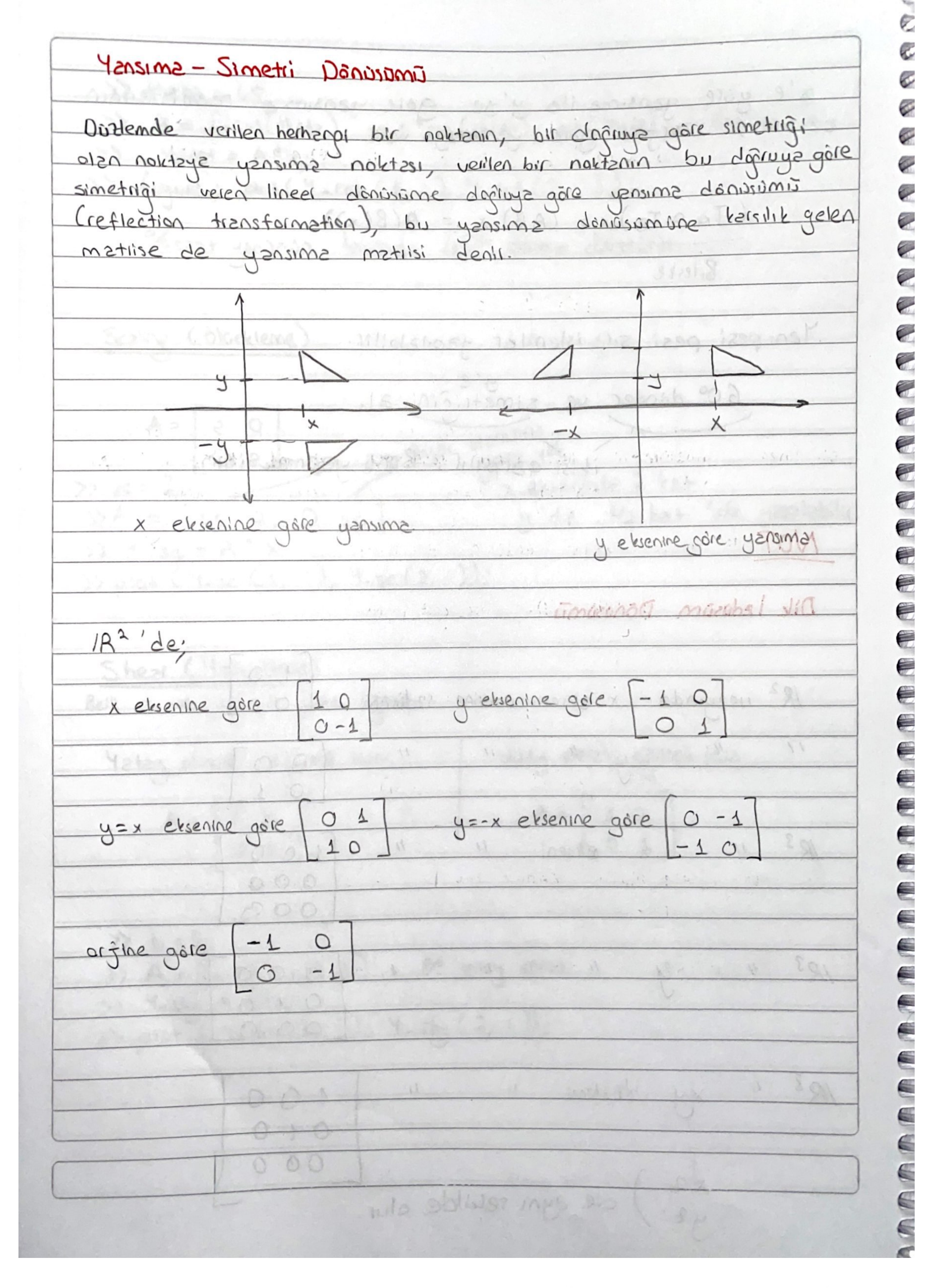
$\mathbb{R}^3$ " y " " " $\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$

$\mathbb{R}^3$ " xy düzlemi " " $\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$

xz, yz) de aynı şekilde olur.

Yansıma - Simetri Dönüşümü

Düzlemde verilen herhangi bir noktanın, bir doğruya göre simetriği olan noktaya yansıma noktası, verilen bir noktanın bu doğruya göre simetriği veren lineer dönüşüme doğruya göre yansıma dönüşümü (reflection transformation), bu yansıma dönüşümüne karşılık gelen matrise de yansıma matrisi denir.



x eksenine göre yansıma

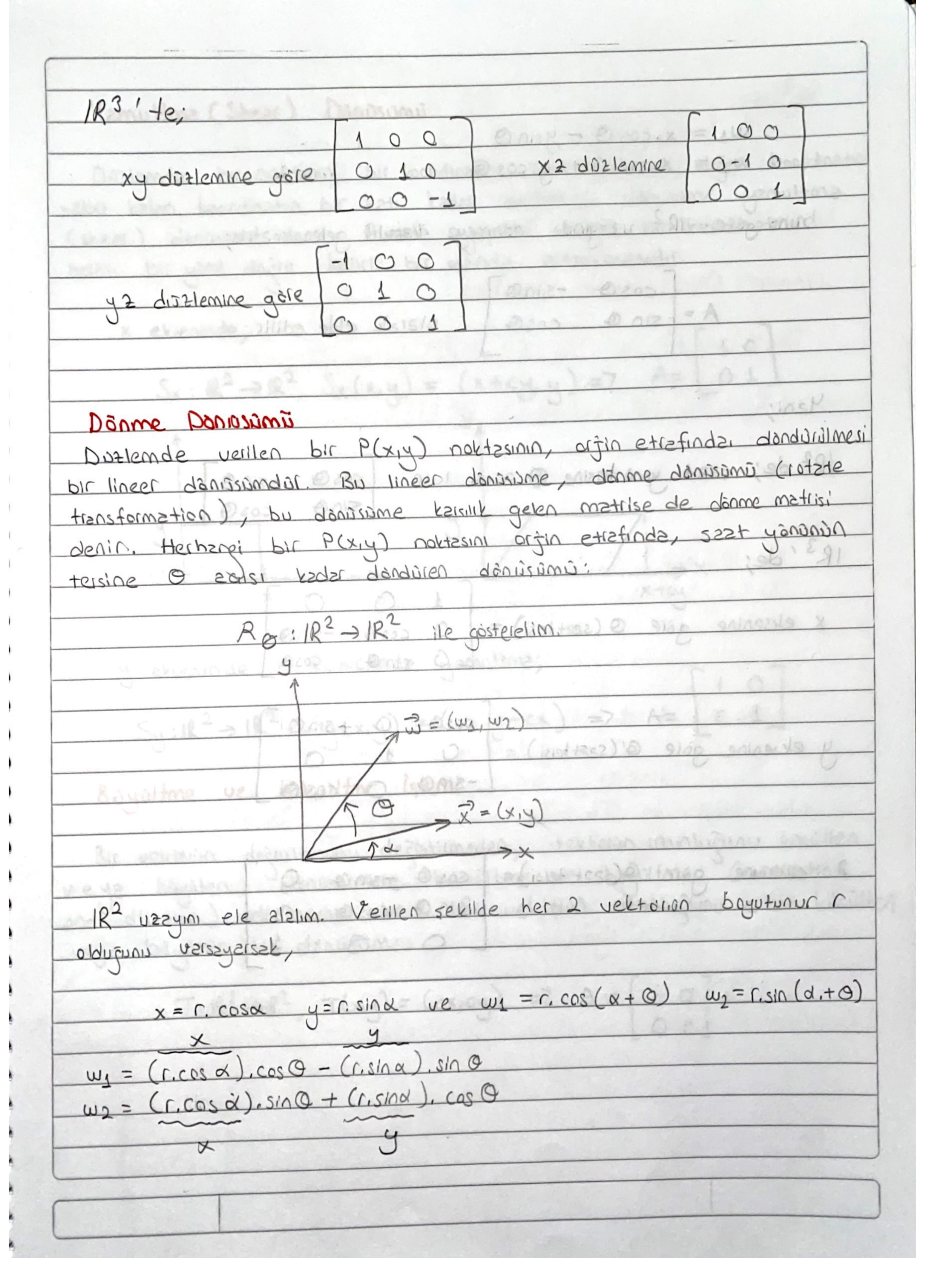
y eksenine göre yansıma

$\mathbb{R}^2$'de;

x eksenine göre $\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$ y eksenine göre $\begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$

$y=x$ eksenine göre $\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$ $y=-x$ eksenine göre $\begin{bmatrix} 0 & -1 \\ -1 & 0 \end{bmatrix}$

orjine göre $\begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}$



$\mathbb{R}^3$'te;

xy düzlemine göre $\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{bmatrix}$ xz düzlemine $\begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

yz düzlemine göre $\begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

Dönme Dönüşümü

Düzlemde verilen bir $P(x,y)$ noktasının, orjin etrafında döndürülmesi bir lineer dönüşümdür. Bu lineer dönüşüme, dönme dönüşümü (rotate transformation), bu dönüşüme karşılık gelen matrise de dönme matrisi denir. Herhangi bir $P(x,y)$ noktasını orjin etrafında, saat yönünün tersine θ açısı kadar döndüren dönüşümü:

$R_\theta : \mathbb{R}^2 \to \mathbb{R}^2$ ile gösterelim.

$\mathbb{R}^2$ uzayını ele alalım. Verilen şekilde her 2 vektörün boyutunun r olduğunu varsayarsak,

$x = r.\cos\alpha \quad y = r.\sin\alpha \quad$ ve $\quad w_1 = r.\cos(\alpha+\theta) \quad w_2 = r.\sin(\alpha+\theta)$

$w_1 = \underbrace{(r.\cos\alpha)}_{x}.\cos\theta - \underbrace{(r.\sin\alpha)}_{y}.\sin\theta$

$w_2 = \underbrace{(r.\cos\alpha)}_{x}.\sin\theta + \underbrace{(r.\sin\alpha)}_{y}.\cos\theta$

$$w_1 = x.\cos\theta - y\sin\theta$$
$$w_2 = x.\sin\theta + y.\cos\theta$$

buna göre IR^2 uzayında dönmeye karşılık gelen matris;

$$A = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$ olarak elde edilir.

Yani;

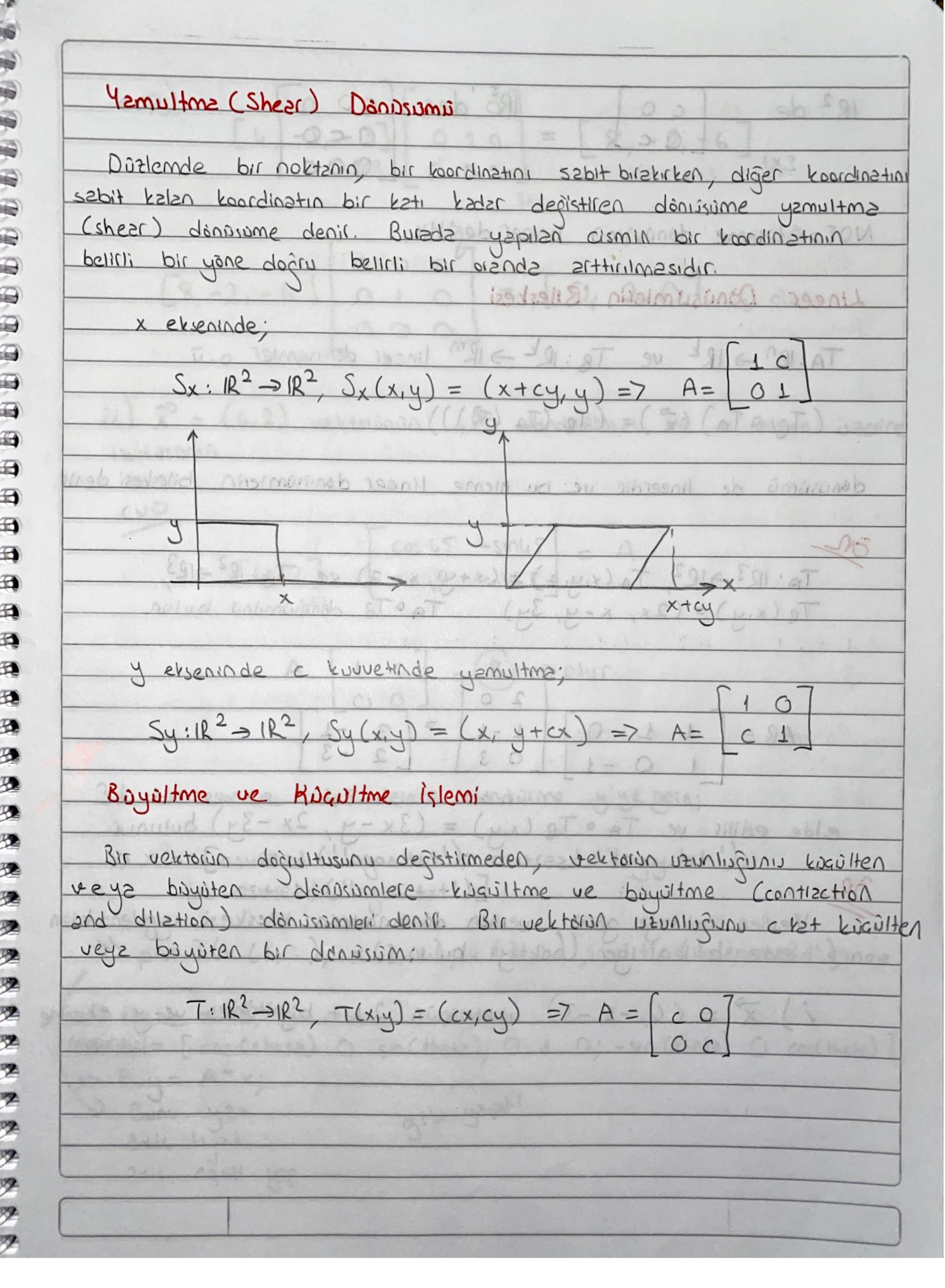
IR^2 de; saat yönü tersine θ açısı için $\begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$

IR^3'de;

x eksenine göre θ (saat tersi) $= \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix}$

y eksenine göre θ (saat tersi) $= \begin{bmatrix} \cos\theta & 0 & +\sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix}$

z eksenine göre θ (saat tersi) $= \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$



Yamultma (Shear) Dönüşümü

Düzlemde bir noktanın, bir koordinatını sabit bırakırken, diğer koordinatını sabit kalan koordinatın bir katı kadar değiştiren dönüşüme yamultma (shear) dönüşüme denir. Burada yapılan cismin bir koordinatının belirli bir yöne doğru belirli bir oranda arttırılmasıdır.

x ekseninde;

$$S_x : \mathbb{R}^2 \to \mathbb{R}^2,\quad S_x(x,y) = (x+cy,\ y) \Rightarrow A = \begin{bmatrix} 1 & c \\ 0 & 1 \end{bmatrix}$$

y ekseninde c kuvvetinde yamultma;

$$S_y : \mathbb{R}^2 \to \mathbb{R}^2,\quad S_y(x,y) = (x,\ y+cx) \Rightarrow A = \begin{bmatrix} 1 & 0 \\ c & 1 \end{bmatrix}$$

Büyültme ve Küçültme İşlemi

Bir vektörün doğrultusunu değiştirmeden, vektörün uzunluğunu küçülten veya büyüten dönüşümlere küçültme ve büyültme (contraction and dilation) dönüşümleri denir. Bir vektörün uzunluğunu c kat küçülten veya büyüten bir dönüşüm;

$$T : \mathbb{R}^2 \to \mathbb{R}^2,\quad T(x,y) = (cx,\ cy) \Rightarrow A = \begin{bmatrix} c & 0 \\ 0 & c \end{bmatrix}$$

$\mathbb{R}^2$ de $\begin{bmatrix} c & 0 \\ 0 & c \end{bmatrix}$ $\mathbb{R}^3$ 'de $\begin{bmatrix} c & 0 & 0 \\ 0 & c & 0 \\ 0 & 0 & c \end{bmatrix}$

NOT = Öteleme dönüşümü lineer değildir.

Lineer Dönüşümlerin Bileşkesi

$T_A : \mathbb{R}^n \to \mathbb{R}^k$ ve $T_B : \mathbb{R}^k \to \mathbb{R}^m$ lineer dönüşümler o.ü

$(T_B \circ T_A)(\vec{x}) = (T_B(T_A(\vec{x})))$

dönüşümü de lineerdir ve bu işleme lineer dönüşümlerin bileşkesi denir.

ÖR

$T_A : \mathbb{R}^3 \to \mathbb{R}^2$, $T_A(x,y,z) = (x+y, x-z)$ ve $T_B : \mathbb{R}^2 \to \mathbb{R}^3$, $T_B(x,y) = (2x, x-y, 3y)$ $T_A \circ T_B$ dönüşümünü bulun.

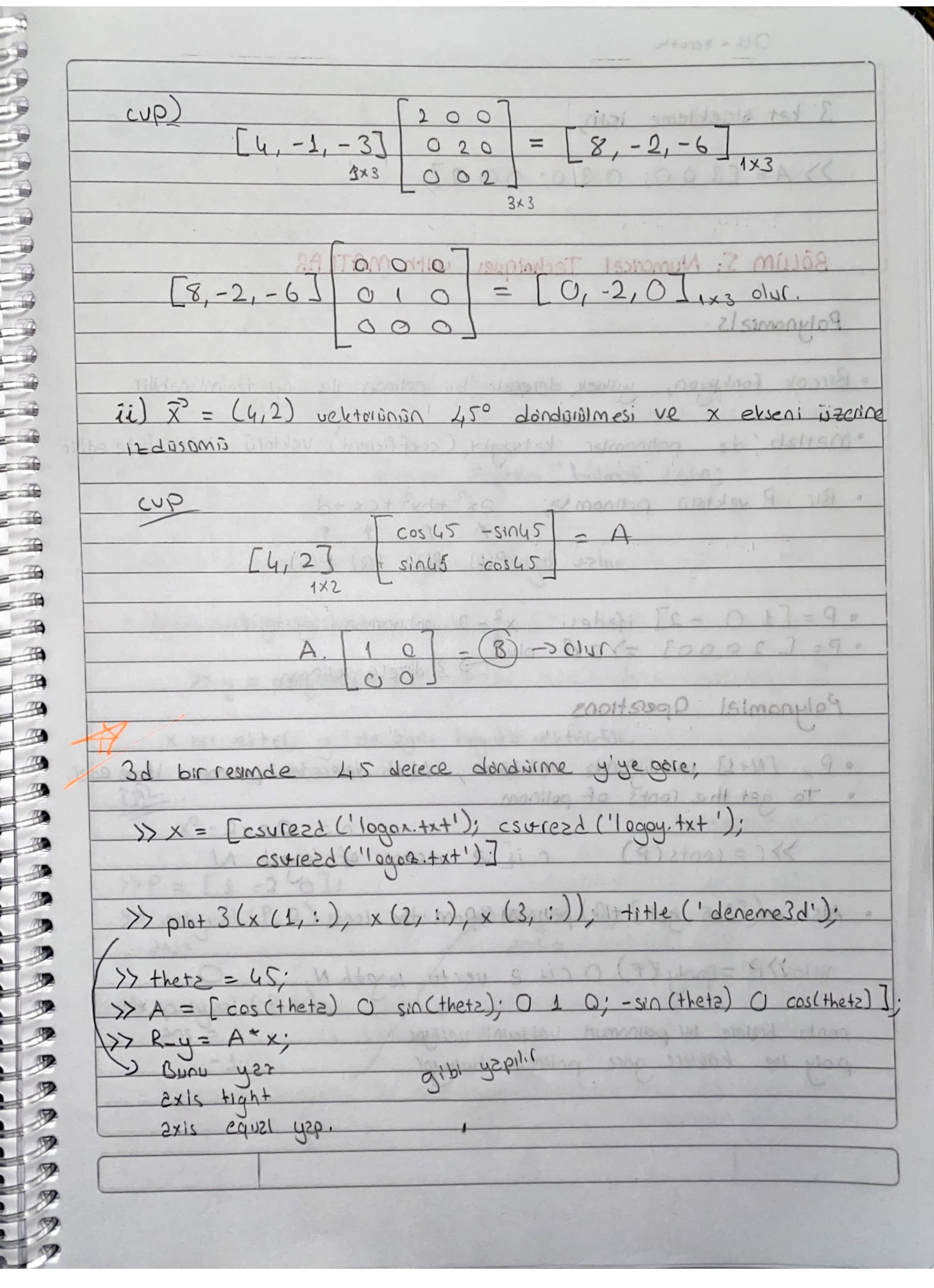
$$AB = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & -1 \end{bmatrix} \begin{bmatrix} 2 & 0 \\ 1 & -1 \\ 0 & 3 \end{bmatrix} = \begin{bmatrix} 3 & -1 \\ 2 & -3 \end{bmatrix}$$

elde edilir ve $T_A \circ T_B(x,y) = (3x-y, 2x-3y)$ bulunur.

ÖR

Verilen noktaya göre, verilen dönüşümlerin bileşkesi uygulandıktan sonra elde edilen yeni noktayı bulun.

i) $\vec{x} = (4, -1, -3)$ vektörünün 2 kat büyütülmesi ve y ekenine izdüşümü



cvp)

$$[4,-1,-3]_{1\times3}\begin{bmatrix}2&0&0\\0&2&0\\0&0&2\end{bmatrix}_{3\times3}=[8,-2,-6]_{1\times3}$$

$$[8,-2,-6]\begin{bmatrix}0&0&0\\0&1&0\\0&0&0\end{bmatrix}=[0,-2,0]_{1\times3} \text{ olur.}$$

ii) $\vec{x}=(4,2)$ vektörünün $45°$ döndürülmesi ve x ekseni üzerine izdüşümü

cvp

$$[4,2]_{1\times2}\begin{bmatrix}\cos45 & -\sin45\\ \sin45 & \cos45\end{bmatrix}=A$$

$$A.\begin{bmatrix}1&0\\0&0\end{bmatrix}=(B)\rightarrow \text{olur}$$

3d bir resimde 45 derece döndürme y'ye göre;

```
>> x = [csvread('logox.txt'); csvread('logoy.txt');
        csvread('logoz.txt')]
>> plot3(x(1,:), x(2,:), x(3,:)), title('deneme3d');
>> theta = 45;
>> A = [cos(theta) 0 sin(theta); 0 1 0; -sin(theta) 0 cos(theta)];
>> R_y = A*x;
```

→ Bunu yaz
axis tight
axis equal yap.

gibi yapılır

0th = zeroth

3 kat ölçekleme için;

```
>> A= [3 0 0; 0 3 0; 0 0 3];
```

BÖLÜM 5: Numerical Techniques with MATLAB

Polynomials

- Birçok fonksiyon, yüksek dereceli bir polinom ile iyi tanımlanabilir.
- Matlab'da polinomlar katsayılar (coefficient) vektörü olarak ifade edilir.
- Bir P vektörü polinomsa; $ax^3 + bx^2 + cx + d$
  - $a \leftarrow P(1)$, $b \leftarrow P(2)$, $c \leftarrow P(3)$, $d \leftarrow P(4)$

- P = [1 0 -2] ifadesi $x^2 - 2$ polinomunu gösterir.
- P = [2 0 0 0] => $2x^3$ olur. → 3. derece polinom

Polynomial Operations

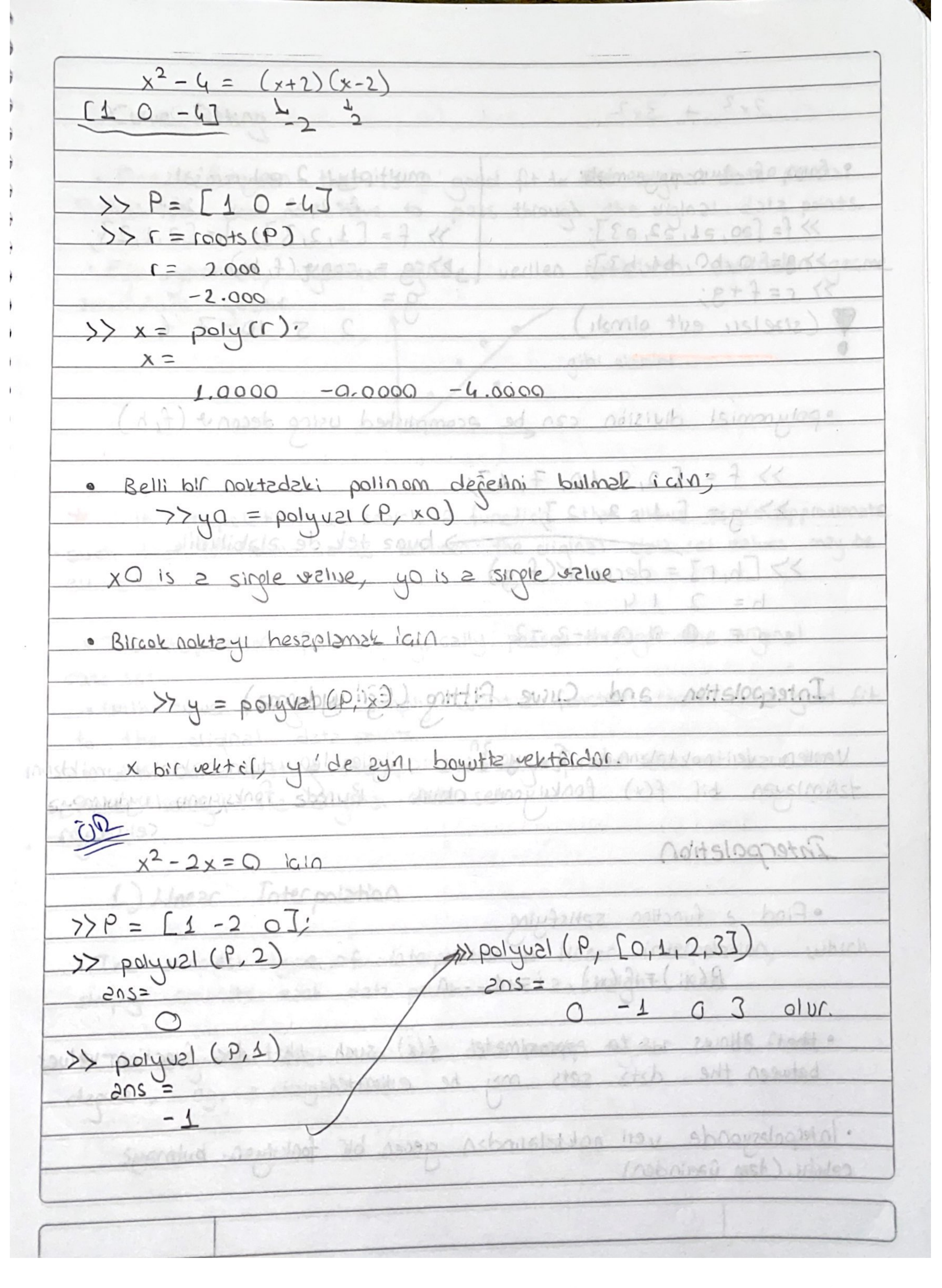
- P, [N+1] uzunluğunda vektörse, N.th dereceden polinomu ifade eder.
- To get the roots of polinom

```
>> r = roots(P)
```
r is a vector of length N

- Also we can get the polinom from the roots

```
>> P = poly(r)
```
r is a vector length N.

roots bizlere bir polinomun köklerini veriyor.
poly ise köklere göre polinom buluyor.



$x^2 - 4 = (x+2)(x-2)$

[1 0 -4] → -2, 2

```
>> P = [1 0 -4]
>> r = roots(P)
   r = 2.000
      -2.000
>> x = poly(r).
   x =
       1.0000   -0.0000   -4.0000
```

- Belli bir noktadaki polinom değerini bulmak için;

```
>> y0 = polyval(P, x0)
```

x0 is a single value, y0 is a single value.

- Birçok noktayı hesaplamak için

```
>> y = polyval(P, x)
```

x bir vektör, y'de aynı boyutta vektördür.

Ör

$x^2 - 2x = 0$ için

```
>> P = [1 -2 0];
>> polyval(P, 2)
 ans =
     0
>> polyval(P, 1)
 ans =
     -1
>> polyval(P, [0,1,2,3])
 ans =
     0  -1  0  3   olur.
```

$2x^2 + 3x^2$

- Sum of two polynomials:

```
>> f= [a0, a1, a2, a3];
>> g=[0, b0, b1, b3];
>> r=f+g;
```

(sizeları eşit olmalı)

multiply 2 polynomials

$x^2 + 2x + 3$ $\quad$ $2x^2 + x + 2$

```
>> f = [1,2,3]; h=[2,1,2];
>> g = conv(f, h)
g =
     2  5  10  7  6
```

- polynomial division can be ecomplished using deconv(f, h)

```
>> f = [2, 5, 10, 7, 6];
>> g= [1, 2, 2];
>> [h,r] = deconv(f,g)
h =  2  1  4
r =  0  0  0  -3  -2
```

→ buna tek de alabilirdik

Interpolation and Curve Fitting (Eğri uydurma)

Verilen veri noktalarında $\{x_i, y_i\}_{i=1}^{n}$ ilgilenilen tüm noktaların miktarını tanımlayan bir $f(x)$ fonksiyonu olsun. Burada fonksiyonu uydurmaya çalışacağız.

Interpolation

- Find a function satisfying

$$P(x_i) = f(x_i), \quad i = 1 \ldots n$$

- that allows us to approximate $f(x)$ such that the function values between the data sets may be estimated.

- Interpolasyonda veri noktalarından geçen bir fonksiyon bulmaya çalışılır. (tam üzerinden)

Curve Fitting

• Find a function that is a good fit to the original data points. The function does not have to pass through the original data points.

• Eğri uydurmada, uydurulan eğride, verilen noktaların üzerinden geçme zorunluluğu yoktur.

gibi olabilir.

★ With interpolation we search a function that allows us to approximate such that functional values between the original data set values may be estimated.

• The interpolation function typically passes through the original data set.

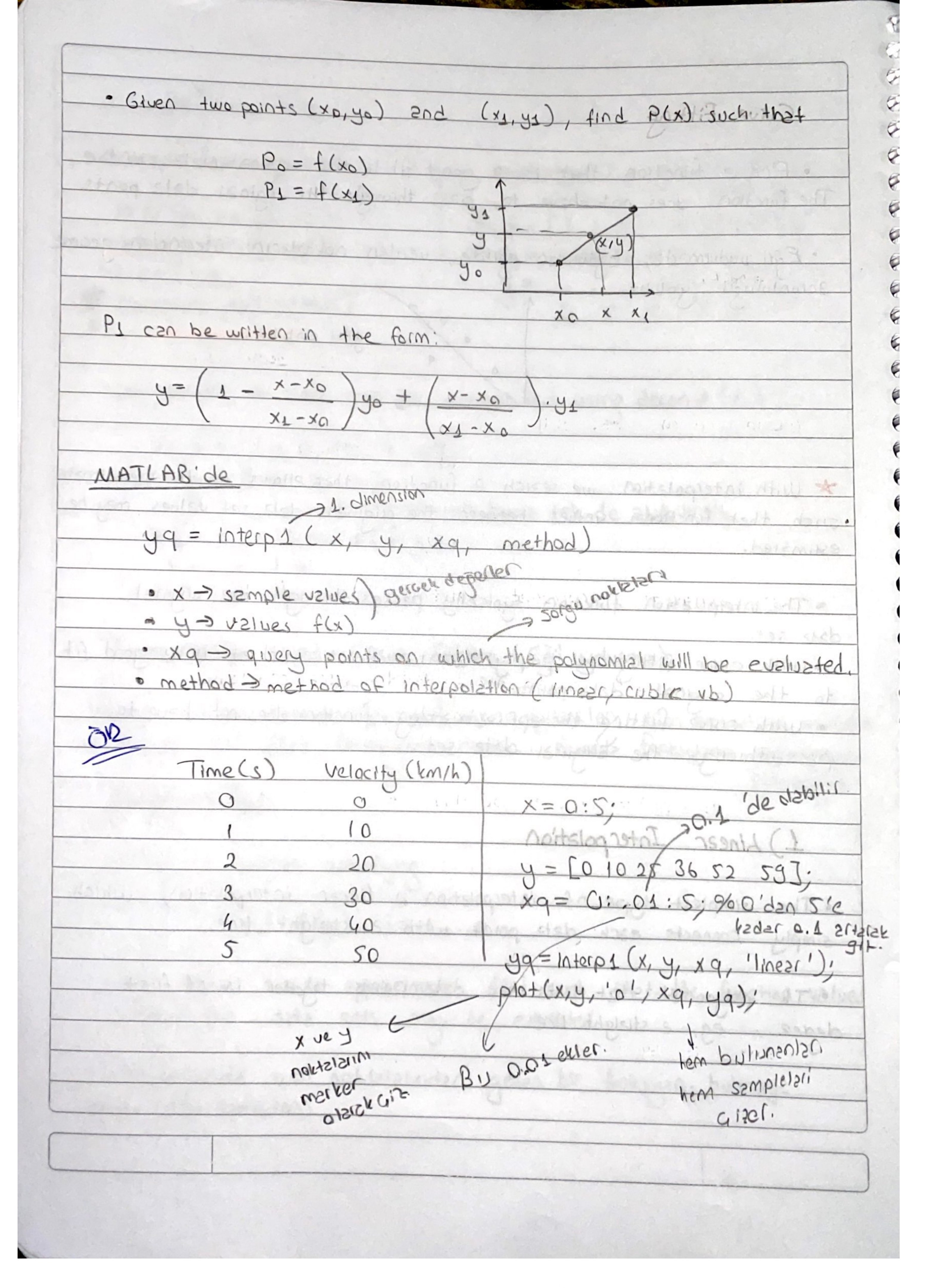
• With curve fitting we simply want a function that is a good fit to the original data points.

• With curve fitting the approximating function does not have to pass through the original data set.

1) Linear Interpolation

• The simplest type of interpolation is linear interpolation, which simply connects each data point with a straight line.

• The polynomial that links the data points together is of first degree, eg, a straight line.



- Given two points (x_0, y_0) and (x_1, y_1), find $P(x)$ such that

$$P_0 = f(x_0)$$
$$P_1 = f(x_1)$$

P_1 can be written in the form:

$$y = \left(1 - \frac{x - x_0}{x_1 - x_0}\right) y_0 + \left(\frac{x - x_0}{x_1 - x_0}\right) \cdot y_1$$

MATLAB'de

```
yq = interp1 (x, y, xq, method)
```

(interp1 → 1. dimension)

- x → sample values } gerçek değerler
- y → values f(x)
- xq → query points on which the polynomial will be evaluated. (sorgu noktaları)
- method → method of interpolation (linear, cubic vb)

ÖR

Time (s)	Velocity (km/h)
0	0
1	10
2	20
3	30
4	40
5	50

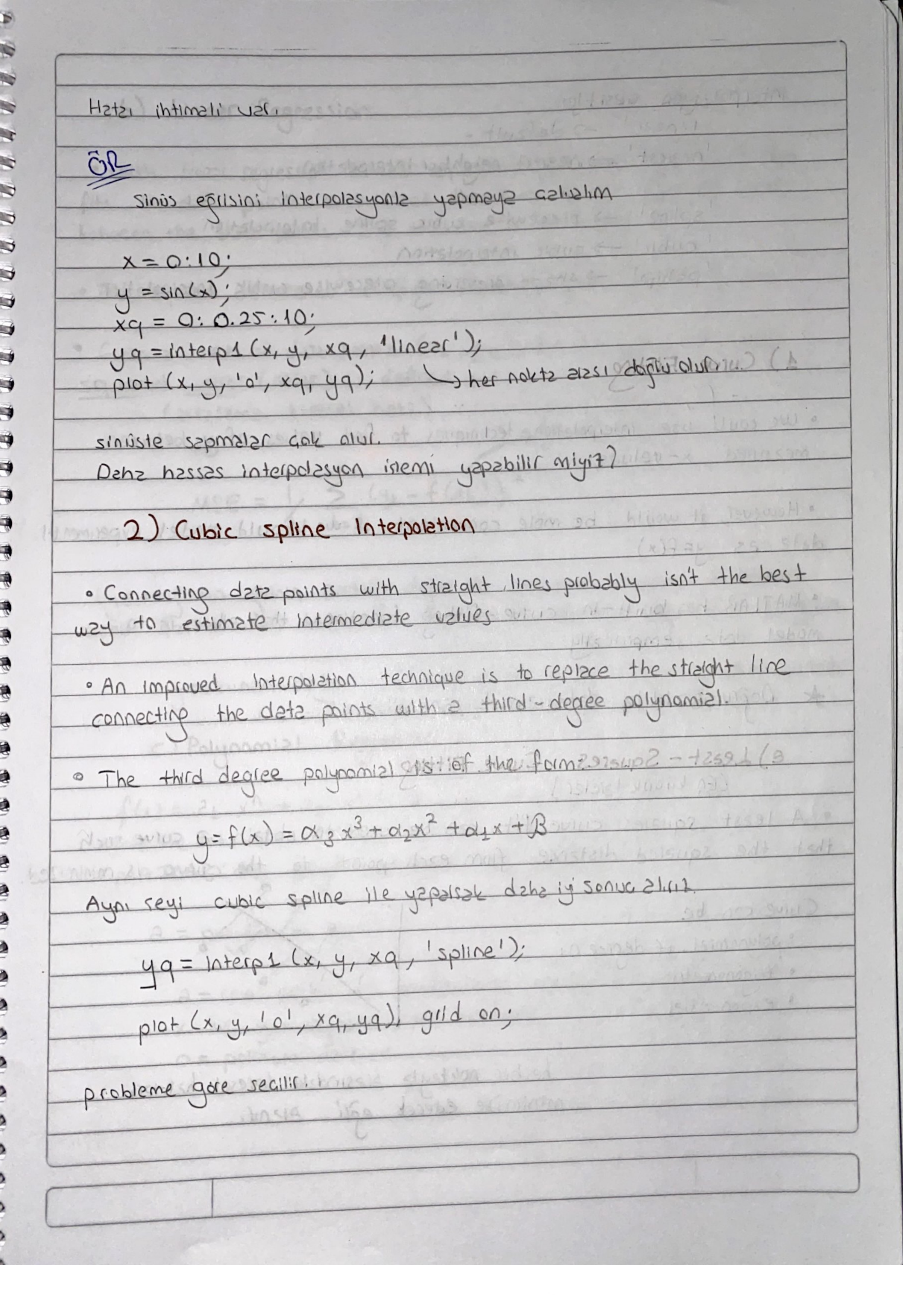
```
x = 0:5;
y = [0 10 25 36 52 59];
xq = 0:.01:5; % 0'dan 5'e kadar 0.1 artarak git.
yq = interp1 (x, y, xq, 'linear');
plot(x,y,'o', xq, yq);
```

(.01 → 0.1 'de olabilir)

Bu 0.01 ekler.

'o' → x ve y noktalarını marker olarak çiz.

xq, yq → hem bulunanları hem sampleları çizer.



Hatalı ihtimali var.

ÖR

Sinüs eğrisini interpolasyonla yapmaya çalışalım

```
x = 0:10;
y = sin(x);
xq = 0:0.25:10;
yq = interp1(x, y, xq, 'linear');
plot(x, y, 'o', xq, yq);
```

↳ her nokta arası doğru olur.

sinüste sapmalar çok olur.
Daha hassas interpolasyon işlemi yapabilir miyiz?

2) Cubic spline Interpolation

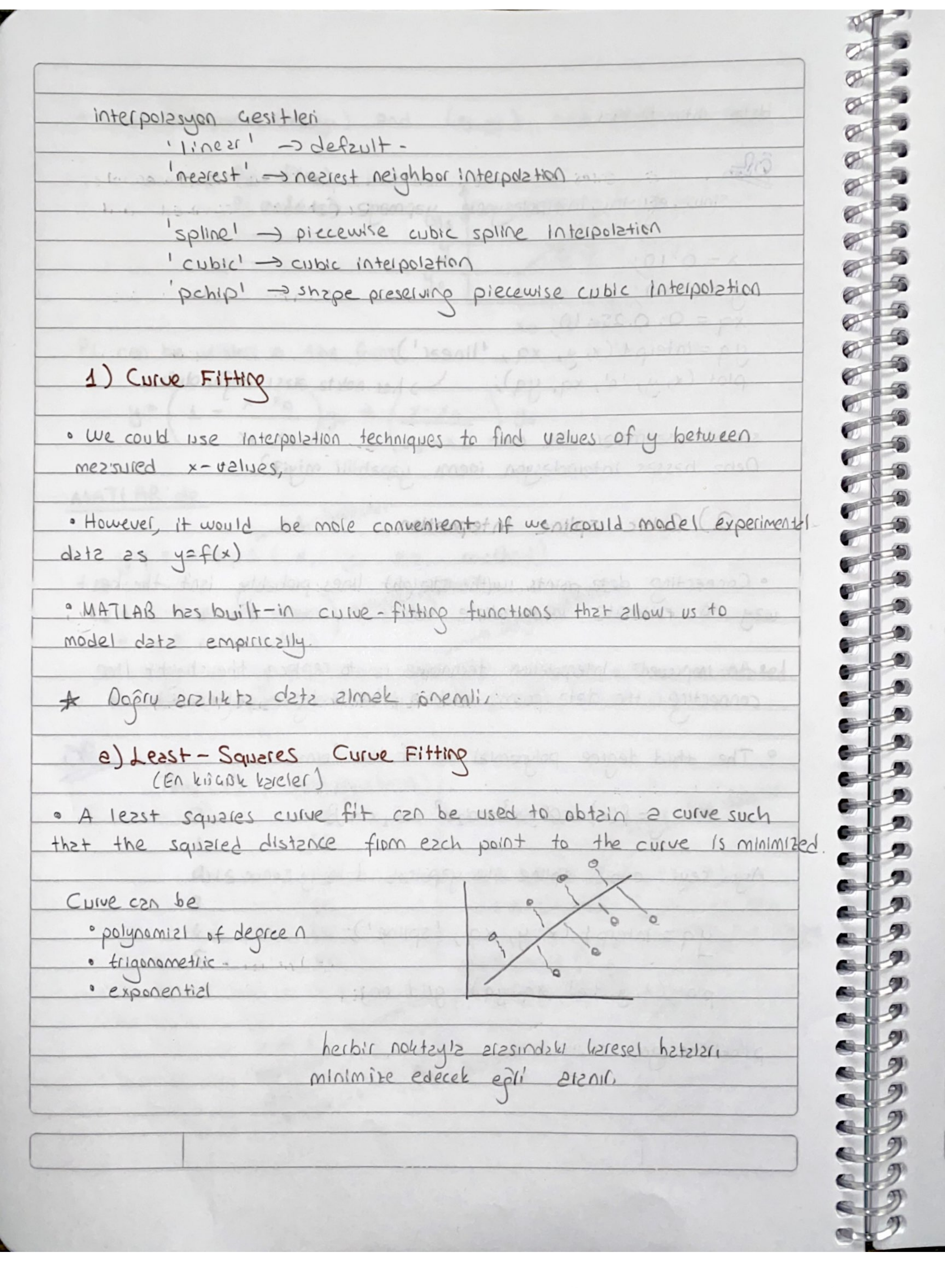
- Connecting data points with straight lines probably isn't the best way to estimate intermediate values
- An improved interpolation technique is to replace the straight line connecting the data points with a third-degree polynomial.
- The third degree polynomial is of the form

$$y = f(x) = \alpha_3 x^3 + \alpha_2 x^2 + \alpha_1 x + \beta$$

Aynı şeyi cubic spline ile yapalsak daha iyi sonuç alırız.

```
yq = interp1(x, y, xq, 'spline');
plot(x, y, 'o', xq, yq); grid on;
```

probleme göre seçilir.



interpolasyon çeşitleri

'linear' → default.

'nearest' → nearest neighbor interpolation

'spline' → piecewise cubic spline interpolation

'cubic' → cubic interpolation

'pchip' → shape preserving piecewise cubic interpolation

1) Curve Fitting

- We could use interpolation techniques to find values of y between measured x-values,
- However, it would be more convenient if we could model experimental data as $y=f(x)$
- MATLAB has built-in curve-fitting functions that allow us to model data empirically.

★ Doğru aralıkta data almak önemli.

a) Least-Squares Curve Fitting
(En küçük kareler)

- A least squares curve fit can be used to obtain a curve such that the squared distance from each point to the curve is minimized.

Curve can be
- polynomial of degree n
- trigonometric
- exponential

herbir noktayla arasındaki karesel hataları minimize edecek eğri aranır.

b) Linear Regression

- With linear regression a linear equation, $y = mx + b$, is chosen that fits the data points such that the sum of the squared error between the data points and the line is minimized.

- The squared distance is computed with respect to the y axis.

- Given a set of data points (x_k, y_k), $k = 1 ... N$, the mean squared error (MSE) is defined as:
  (ortalama karesel hata)

$$MSE = \frac{1}{N} \sum_{k=1}^{N} (y_k - f(x_k))^2$$

→ $f(x_k) = (m x_k + b)$

Uydurduğumuz eğrinin ürettiği değer ile gerçekte okumuş olduğumuz verinin farklarının karelerinin toplamını minimize etmeye çalışıyoruz.

y = okunan data   f(x) = eğrinin o nokta için uydurduğu datadır.
arasındaki fark minimum olsun istiyoruz.

c) Polynomial Regression

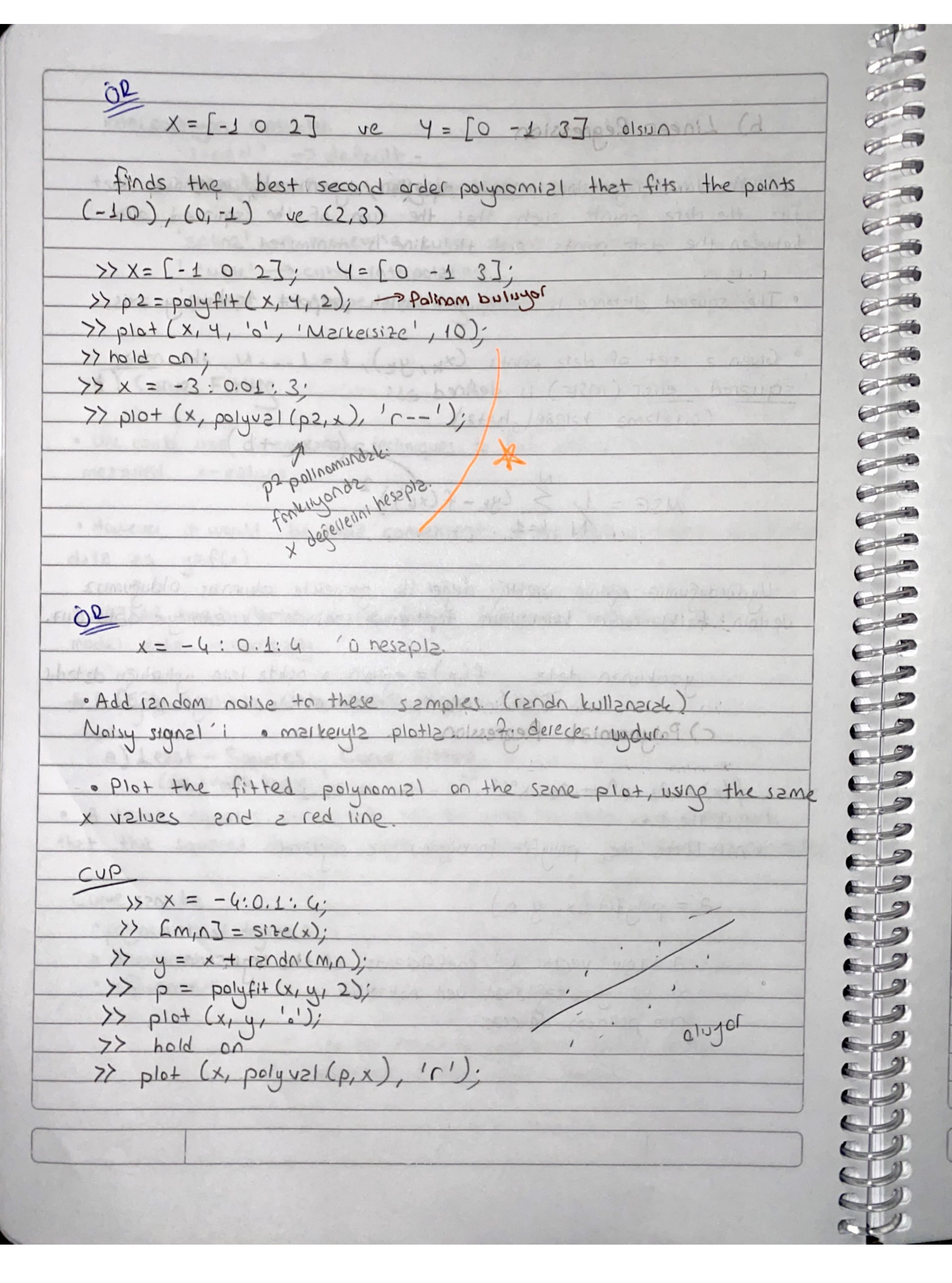
$$f(x) = a_1 . x^n + a_2 . x^{n-1} + --- + a_{n-1}$$

fits the data

- MATLAB'de, polyfit fonksiyonu ile sağlanır

a = polyfit (x, y, n)

a = row vector of coefficients of the regression
x ve y = sağlanan veri noktaları
n = polinom derecesi



ÖR

x = [-1 0 2] ve y = [0 -1 3] olsun

finds the best second order polynomial that fits the points (-1,0), (0,-1) ve (2,3)

```
>> x=[-1 0 2]; y=[0 -1 3];
>> p2=polyfit(x,y,2);   → Polinom buluyor
>> plot(x,y,'o','Markersize',10);
>> hold on;
>> x = -3:0.01:3;
>> plot(x,polyval(p2,x),'r--');
```

↑ p2 polinomundaki fonksiyonda x değerlerini hesapla.

ÖR

x = -4:0.1:4 'ü hesapla.

- Add random noise to these samples. (randn kullanarak)

Noisy signal'i • markeryla plotla 2. derecesini uydur.

- Plot the fitted polynomial on the same plot, using the same x values and a red line.

CVP

```
>> x = -4:0.1:4;
>> [m,n] = size(x);
>> y = x + randn(m,n);
>> p = polyfit(x,y,2);
>> plot(x,y,'o');
>> hold on
>> plot(x,polyval(p,x),'r');
```

oluyor

ÖR

```
>> x = 0:5;
>> y = [0 10 25 36 52 59];

>> x_ls = 0:0.01:5;
```

Lineer fitting → bitişik de olabilir. → square (kare)

```
>> figure; subplot (2 2 1); plot (x, y, 's'); hold on;

>> p1 = polyfit (x, y, 1);

>> y1 = polyval (p1, x_ls);

>> plot (x_ls, y1);

>> axis ([0 5 0 60]); grid on;
>> title ('First Degree');
```

second degree fitting

```
>> p2 = polyfit (x, y, 2);
>> y2 = polyval (p2, x_ls);
>> subplot (2 2 2); plot (x, y, 's'); hold on;
>> plot (x_ls, y2);
>> title ('Second Degree');
```

fifth degree

```
>> p5 = polyfit (x, y, 5)
>> y5 = polyval (p5, x_ls);
>> subplot (2 2 3); plot (x, y, 's'); hold on;
>> plot (x_ls, y5);
>> title ('Fifth Degree');
```

5. derece daha smooth daha iyi oldu.

10. derecede ise subplot (2 2 4) için → mantıksız oldu. Her zaman yüksek dereceden eğri iyi olmaz.

yani datayı tanımak gerek. En fit dereceli eğriyi bulmalıyız.

Interpolation in 2D

vq = interp2 (x, y, v, xq, yq, method)
↓
2 boyutlu

(x, y) → 2 boyutlu   v'de bunlara denk gelen değer.

V'yi bilmiyoz.   Farklı tür metotlar kullanılabilir.
- cubic, spline, nearest gibi.

- z = peaks (x, y) evaluates the peaks function at x and y.
- We can use z as data points and visualize interpolation using various methods.

ÖR

```
[x, y] = meshgrid (-4:4);

v = peaks (x,y);
surf (x, y, v);
title ('Original Data');
```

6 Devamı

%% Linear Interpolation

```
>> [xq, yq] = meshgrid(-4:0.25:4);

>> vq = interp2(x, y, v, xq, yq, 'linear');

>> surf(xq, yq, vq);   // Daha yumuşak geçişler oldu.

>> title('Linear interpolasyon');
```

cubic yapınca daha kıvrımlı, spline daha iyi olur.

Image Manipulation

- Starting with a random image im = rand(10,10);
- Interpolate the image using 64 times as many points in each direction.

OR

```
%% Generate Random Image
>> im = rand(10,10);
>> figure; subplot(2,2,1);
>> imshow(im);
>> title('original');
%% Interpolasyon

>> [m,n] = size(im);
>> [x,y] = meshgrid(1:m);
>> [xq,yq] = meshgrid(linspace(1, m, 64*m));

>> im_linear = interp2(x,y,im, xq, yq, 'linear');
>> im_cubic = interp2(x,y,im, xq, yq, 'cubic');
>> im_spline = interp2(x,y,im, xq, yq, 'spline');
```

%% Show images

```
>> subplot(2, 2,2);
>> imshow(Im_linear);
>> title('linear interpolant');

>> subplot(2,2,3);
>> imshow(im_cubic);
>> title('Kubic');

>> subplot(2,2,4);
>> imshow(im_spline);
>> title('Spline');
```

Ör (Sınav Sorusu)

Write a function named checkerboards(nrows, ncols, n) that draws a nxn checkerboard according to the user-specified dimensions. Your function must accept three arguments, nrows specifies the number of rows of the checkerboard, ncols specifies the number of columns and n specifies the number of blocks for each row and column. Your function must display a checkerboard by using imagesc function and greyscale colormap. Your function must display a checkerboard by using imagesc function and a grayscale colormap. Your function must return a 2D matrix that represents the checkerboard.

Mesela (4,4,4) için

```
1 0 1 0
0 1 0 1
1 0 1 0
0 1 0 1
```

4,4,2 için

```
1 1 0 0
1 1 0 0
0 0 1 1
0 0 1 1
```

olacak

Benim cevabım

```
function [board] = checkerBoard3 (nrows, ncols, n)

block_rows = floor (nrows /n);
block_cols = floor (ncols / n);

block = ones (block_rows, block_cols);
nonblock = zeros (black_rows, block_cols);

mat = [block; nonblock]';

flipmat = flip (mat, 2);

newmat = [mat; flipmat];

board = repmat (newmat, nrows / length(newmat), ncols / length (newmat));

imagesc (board);
colormap ('gray');
end
```

BÖLÜM-6 IMAGE PROCESSING WITH MATLAB

Resim nedir?

- In its most general form, an image is a function f from $R^2 \rightarrow R$
- f(x,y) give the intensity of a channel at position (x,y)

```
>> I=rand(256, 256);
>> imshow(I);
```

- Each part of an image is a pixel.

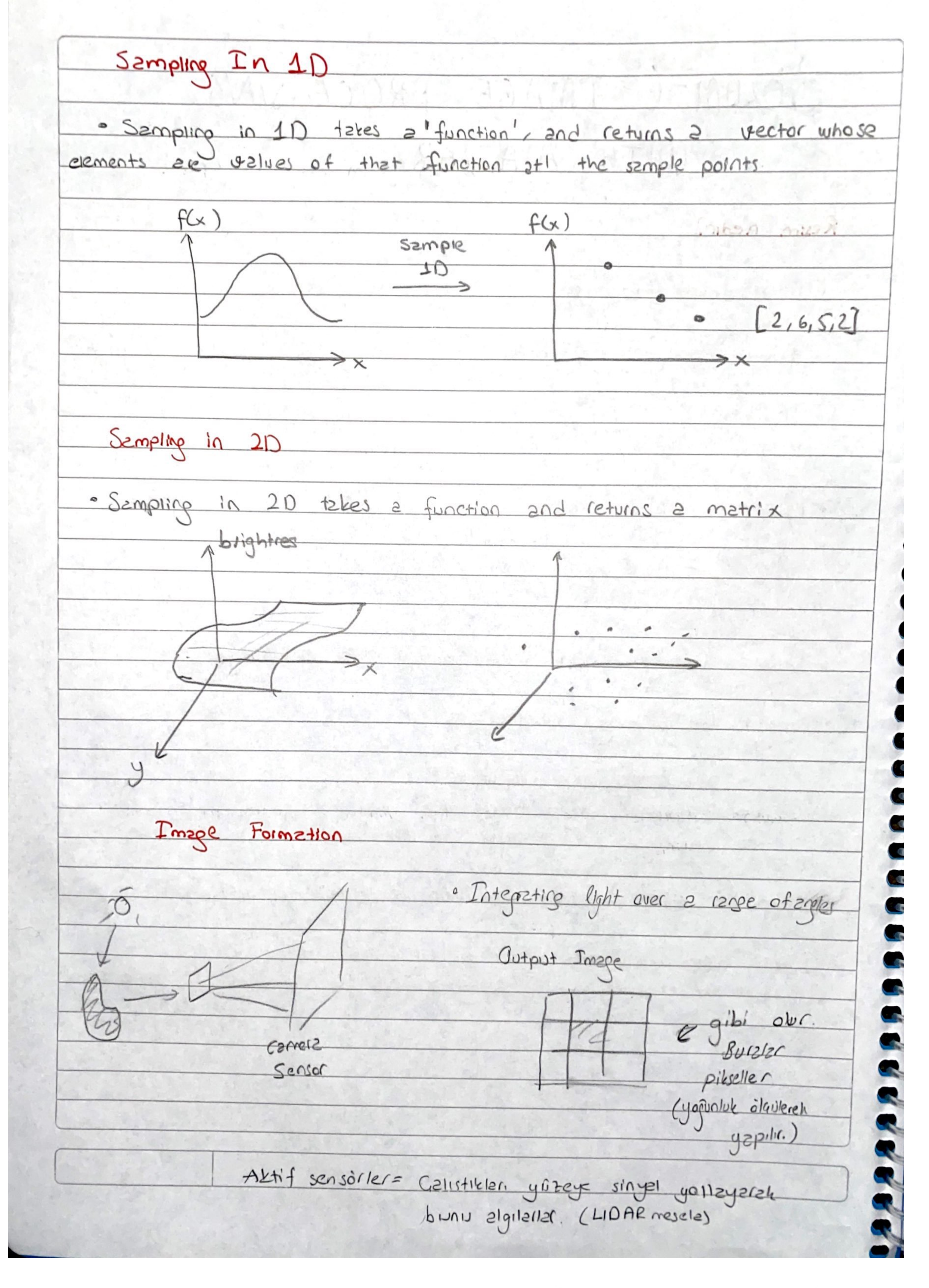
$\rightarrow I(x,y) = 138$

- Signal = fonksiyon bazı fiziksel anlamlı değişkenlere bağlıdır.
- Image = Bu fonksiyon ile örnekleme

 - 2 variables: xy coordinates
 - 3 variables: xy + time (video)
 - 'Brightness' is the value of the function for visible light.

- Can be other physical values too: temperature, pressure, depth.

Görüntü işleme dediğimiz dijital bir veri haline getirilmesi işidir. (ultrasonik, termal, mr ---) (Gama, X-Ray, UV, visible light, IR, Radio waves) (ışık spektrumu)



Sampling In 1D

- Sampling in 1D takes a 'function', and returns a vector whose elements are values of that function at the sample points.

Sampling in 2D

- Sampling in 2D takes a function and returns a matrix

Image Formation

- Integrating light over a range of angles

← gibi olur. Bunlar pikseller (yoğunluk ölçülerek yapılır.)

Aktif sensörler = Çalıştıkları yüzeye sinyal yollayarak bunu algılarlar. (LIDAR mesela)

Resolution

Geometric vs Spatial Resolution

→ En küçük mesafe birimi genelde kare alanlar için kullanılır.

500x500 iki resim olsun

İki resim arasında boyut ve piksel görünümlülüğü önemlidir.

Geometrik → uydudan sensörlerle.

Quantization

Bir resmin Brightness'ı önce az sonra artsın sonra tekrar azalsın.

→ şeklinde olur.

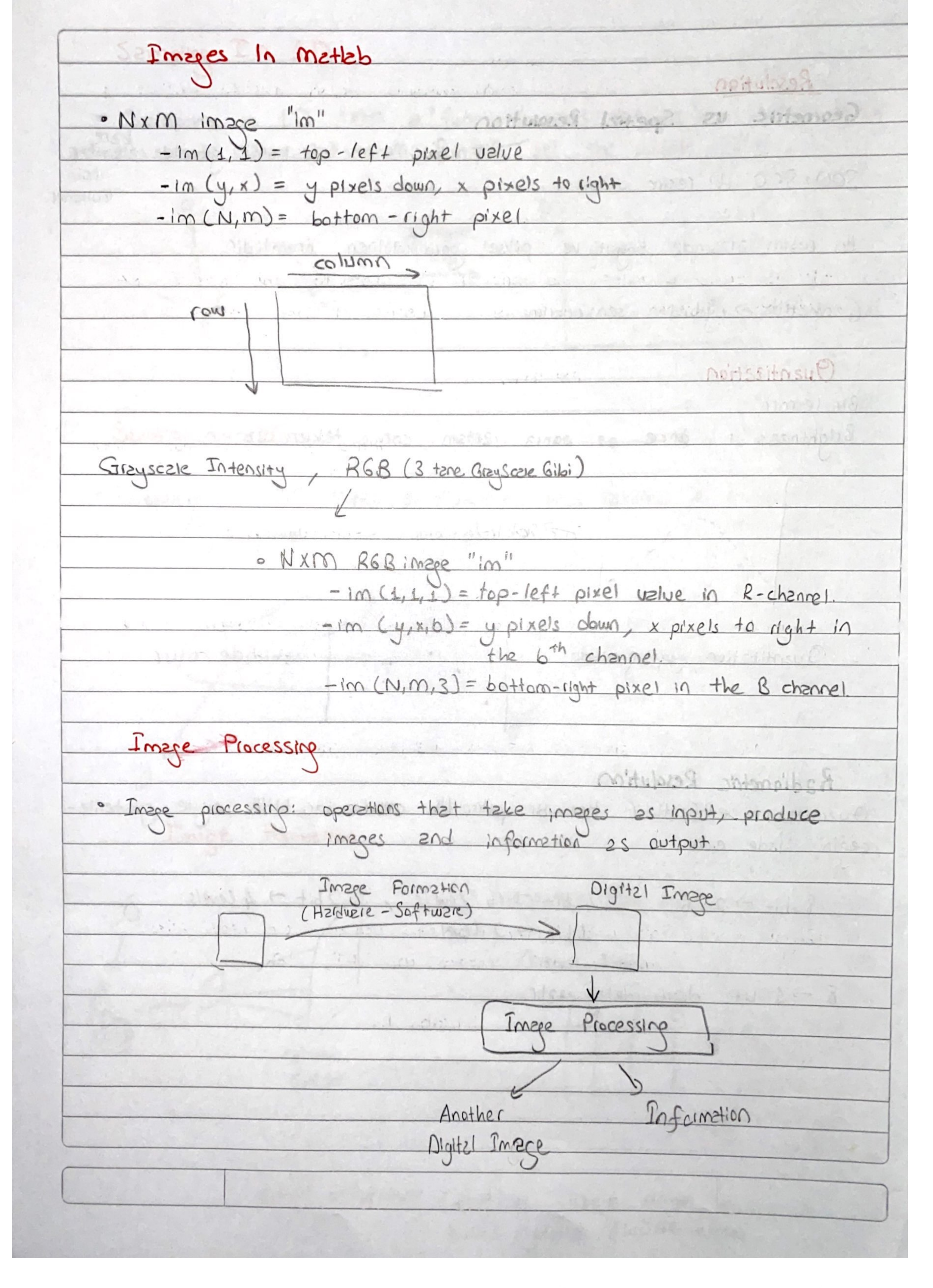
Quantization yaptığımızda şeklinde olur.

Radiometric Resolution

Quantization effectidir. Her bir piksellik verinin kaç bitlik veriyle gösterileceğini ifade eder.

8 bit → 256 , 4 bit → 16 levels , 2 bit → 4 levels
1 bit → 2 levels.

8 → 1 bite doğru detay azalır.



Images In Matlab

- N x M image "im"
  - im(1, 1) = top-left pixel value
  - im(y, x) = y pixels down, x pixels to right
  - im(N, M) = bottom-right pixel

Grayscale Intensity , RGB (3 tane GrayScale Gibi)

- N x M RGB image "im"
  - im(1, 1, 1) = top-left pixel value in R-channel.
  - im(y, x, b) = y pixels down, x pixels to right in the b^th channel.
  - im(N, M, 3) = bottom-right pixel in the B channel.

Image Processing

- Image processing: operations that take images as input, produce images and information as output

carrupt=bozuk

→ mesela uzaklık ölçme gibi

comp. photography

Image Processing, → signal processing, computer vision, graphics, machine learning, statistics, applied math gibi alanlarda kullanılabilir

- The processing of digital images can be divided into several classes:

  - image enhancement = the process of improving the quality and the information content of images so that the results are more suitable for display or further image analysis.
    (görüntü iyileştirme)

  - Image restoration = the operation of taking a corrupt/noisy image and estimating the clean, original image.
    (resimdeki gürültüler vb ortadan kaldırılır.)

  - Image Analysis = The extraction of meaningful information from images
    (Görüntü Analizi)

  - Image compression = type of data compression applied to digital images, to reduce their cost for storage or transmission
    (Görüntü Sıkıştırma)

Computer Vision (Bilgisayarlı Görü) = Amaç, makineye, insan gözüymüş gibi bir algılama kapasitesi kazandırmak.
(öz. insan tespiti)

Images In Matlab

- Reading images: imread open and read the contents of image files in most popular formats (eg. TIFF, JPEG, BMP, GIF and PNG etc)

```
>> im = imread ('faculty.bmp');
```

- Display an image

```
>> figure; imshow (im);
```

• Writing images: imwrite writes image data to the file specified by filename. (kaydetme)

>> imwrite (im, 'faculty.bmp');

Image Viewer Application

• imtool: displays an image and contains several associated tools that can be used to explore the image contents.

>> imtool (im) ← Resim pikselleri hakkında bilgi verir.

Resmin bulunduğu dosya açılır

MATLAB'de

>> im = imread ('faculty.jpg');

im = 485 x 780 x 3 olur → RGB channelları belirtir (uint 8 olarak alıyor.)

(x, y) boyutunda piksel bulunur. 2 boyuttan tek boyuta çekilebilir. Tek RGB yok.

time → tek bir görüntü zamanla değişen görüntüler için kullanılır.

>> imR = im (:, :, 1); // Sadece kırmızı kanalı alır.

>> figure; imshow(imR); // Siyah beyaz ama bu kırmızıyı ifade edir

2 için → G, 3 için B çıkar

→ yüklemesi uzun sürüyor

>> imtool (im); // Resimde gezinirken piksel değerlerini gösteriyor.
// Inspect pixel values'e tıklanarak daha detaylı görünür.

>> imwrite (im, 'faculty.bmp');

Resmi soldan sağa, veya yukarı aşağı çevirmek için

```
>> imlr = fliplr(im);
>> imud = flipud(im);
>> figure; imshow(imud);
>> figure; imshow(imlr);
```

Data Classes and Data Conversions

• To convert an image to a data class and range suitable for image processing, you can use one of the functions listed in the table

Name	Data Class
im2single	single
im2double	double
im2uint8	uint8
im2uint16	uint16
im2int16	int16

• The most common data classes for images are as follows:

- uint 8: 1 byte per pixel, in the [0,255] range
- double: 8 bytes per pixel, [0.0, 1.0] range
- logical: 1 bit per pixel, 1 → white, 0 → black

Bir RGB resmi grayscale'e çekmek için

• rgb2grey → convert an RGB image to grayscale ($\frac{R+G+B}{3}$ her pixel için yapılabilir)

↳ Pratikte böyle olmuyor. Bir ağırlıklı ortalama ile yapılıyor.

```
>> img = rgb2grey(im);
>> figure; imshow(img);
```

(rgb2grey
0.2989 * R + 0.5870 * G + 0.1140 * B)

Görüntü piksellerini double'a çekelim

```
>> imD = im2double(imG); // 0 ile 1 arasına çekilmiş olur.
```

0 ile 255 arasına çekilsin ama double olsun istiyorsak

```
>> imDD = double(imG);    // imDD = imDD + 0.1; yapılabilir.
```

Image Resizing

- imresize(image, scale): returns an image which is scale times of the given image.

```
>> imR1 = imresize(im, 0.5, 'nearest');
>> imR2 = imresize(im, 0.5, 'bilinear');
>> imR3 = imresize(im, 0.5, 'bicubic');
>> figure, subplot(1,3,1), imshow(imR1), title('Nearest-Neighbor Interpolation');
>> subplot(1,3,2), imshow(imR2), title('Bilinear Interpolation');
>> subplot(1,3,3), imshow(imR3), title('Bicubic Interpolation');
```

// bicubic 2 kat büyütünce bilineara göre daha okunaklı oluyor.
// illaki bir kayıp olur.

Image Rotation

- imrotate(image, angle) = rotates image by angle degrees in a counterclockwise direction.
- when you rotate an image, you specify the image to be rotated and the rotation angle, in degrees.
- Like imresize, imrotate allows the user to specify the interpolation method used: nearest-neighbor (default), bilinear, or cubic.

```
>> imRot = imrotate(im, 45, 'bilinear');
```

Döndürme demek aslında olmayan piksel değerlerin ortaya çıkması demek (sazatin tersi).

Image Cropping

- imcrop(image): displays the image in a figure window and creates an Interactive crop Image tool associated with an image.

- imcrop(image, rect) -> crops the image according to the position and dimensions specified in the crop rectangle ([x, y, width, height])

```
>> imCropped = imcrop(im);
>> imCropped = imcrop(im, [40, 40, 200, 200]);
```

↓

bunu şöyle de yapabilirdik

```
>> imCropped = im(40:240, 40:240, :);
```

BÖLÜM-7 IMAGE FILTERING DEVAM

Image Filtering

Compute function of local neighborhood at each position.

$$h(m,n) = \sum_{k,l} f(k,l)\, I(m+k, n+l)$$

- Amacı
  - Enhance images = Denoise, resize, increase contrast, etc.
  - Extract information from images = texture, edges, distinctive points, etc
  - Detect patterns: template matching
  - Deep Convolutional Networks

Avaraj filtre = $\frac{1}{9}\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix}$

filtreleme (konvolüsyon), pencerenin görüntü üzerinde gezdirilmesi işlemidir (Görüntü işleme bkz.)

Bir görüntüye uyguladığımızda alçak frekansları geçirir, yüksek frekansları bastırır.

$\otimes$ → konvolusyon işareti.

$$f[x,y] \otimes g[u,v] = h[x,y]$$

($g[u,v]$ → filtre)

görüntü kenarları kaybedilir. Bunun için çeşitli yöntemler var.

Box Filter

- Pencerenin dahilindeki piksellerin ortalamasını alıp merkez pikselin yeni değerini değiştiriyor. Bu smoothing effecttir. (yumuşatma yani keskin kenarları atar.)

Bütün değerler aynı ağırlıkta

Gaussian Filter

$$\frac{1}{16}\begin{bmatrix}1 & 2 & 1\\ 2 & 4 & 2\\ 1 & 2 & 1\end{bmatrix}$$

→ Yakın komşu daha etkili.

σ (sigma) arttıkça bulanıklık artar.

Gaussian Function

$$g(u,v,\sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{\sigma^2}}$$

yapı korunur ama texture (detaylar) kaybedilir.

Smoothing with Gaussian Filter

- The amount of smoothing (bulanıklaştırma) depends on the value of the spread parameter (σ)

```
for sigma = 1:3:10
    h = fspecial('gaussian', fsize, sigma);
    out = imfilter(im, h);
    imshow(out);
    pause;
end
```

İnsan gözünün bulanıklığı algılaması açısından gaussian daha iyidir.

accentuates = vurgular

Practice with Linear Filters

0	0	0
0	1	0
0	0	0

→ resmin aynısını verir.

0	0	0
0	0	1
0	0	0

→ sola kayar. (Shifted left By 1 pixel)

0	0	0
0	2	0
0	0	0

$- \frac{1}{9}$

1	1	1
1	1	1
1	1	1

$=$

$-1/9$	$-1/9$	$-1/9$
$-1/9$	$2\sim$	$-1/9$
$-1/9$	$-1/9$	$-1/9$

keskinleştirir.

Sharpening Filter

local avarajı ile olan farklılıklar vurgulanır.

Yüksel frekansları geçer Filter

Sobel Filter

yüksek geçiren filtredir.

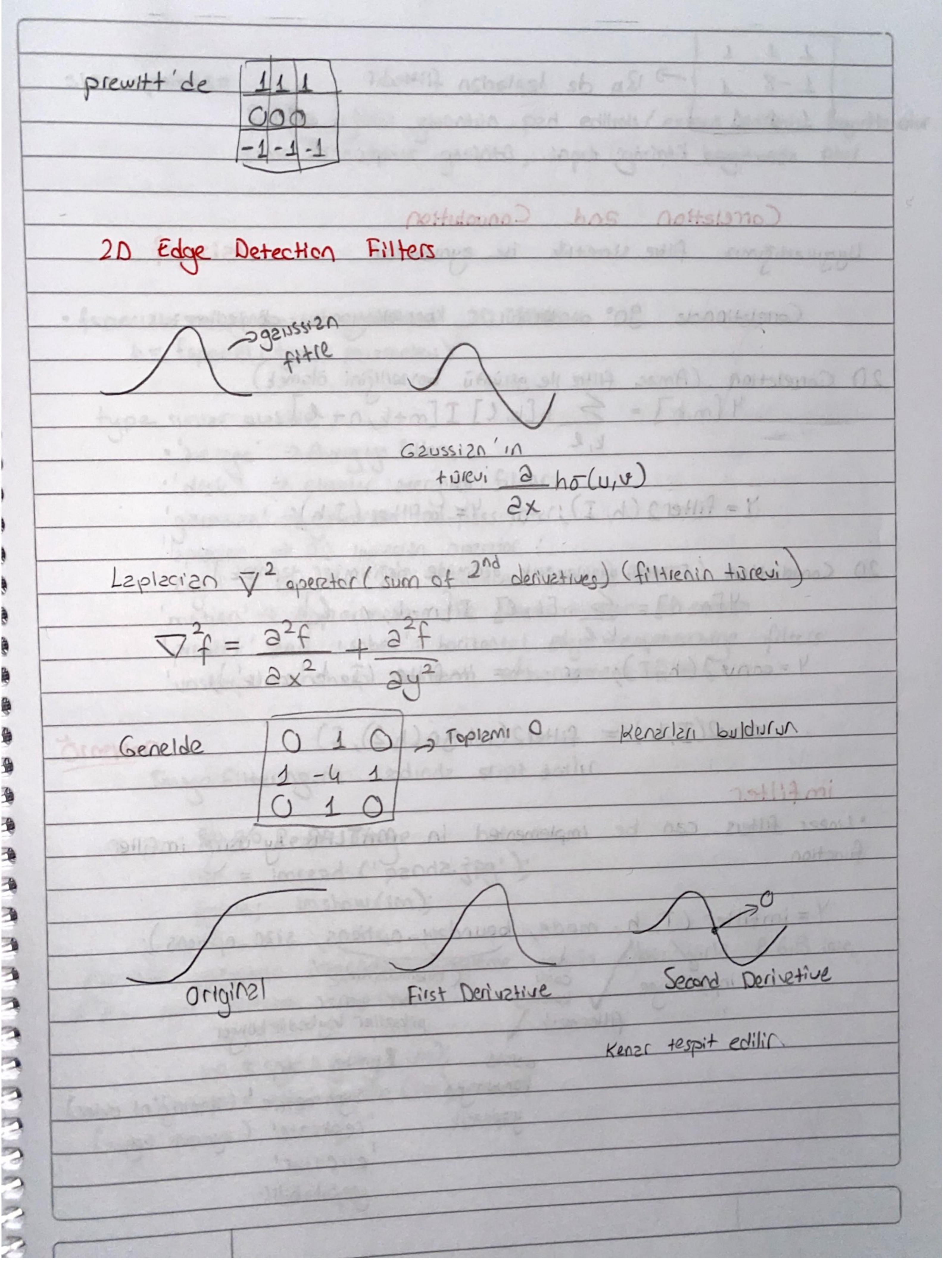
1	0	-1
2	0	-2
1	0	-1

Kenar tespit operatörüdür.

Yatayındaki değişimleri bulduğu için bir dikey kenar tespit operatörüdür.

Yatay tespit için transpozunu alırız

1	2	1
0	0	0
-1	-2	-1



prewitt'de

1	1	1
0	0	0
-1	-1	-1

2D Edge Detection Filters

→ gaussian filtre

Gaussian'ın türevi $\frac{\partial}{\partial x} h_{\sigma}(u,v)$

Laplacian ∇^2 operator (sum of 2^{nd} derivatives) (filtrenin türevi)

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

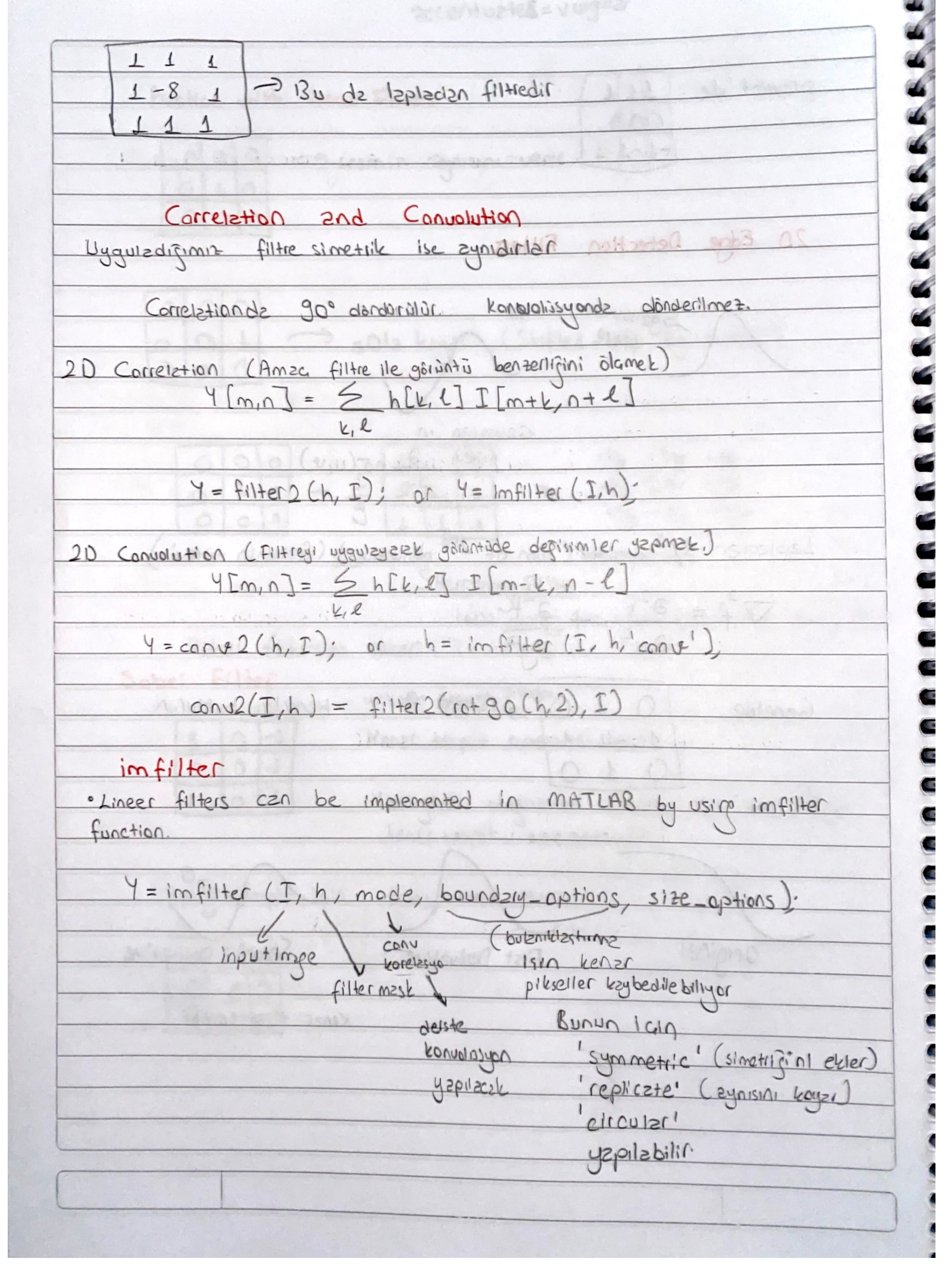
Genelde

0	1	0
1	-4	1
0	1	0

→ Toplamı 0   Kenarları buldurur

Original   First Derivative   Second Derivative → 0

Kenar tespit edilir



1	1	1
1	-8	1
1	1	1

→ Bu da laplacian filtredir

Correlation and Convolution

Uyguladığımız filtre simetrik ise aynıdırlar

Correlationda 90° döndürülür. konvolüsyonda döndürülmez.

2D Correlation (Amaç filtre ile görüntü benzerliğini ölçmek)

$$Y[m,n] = \sum_{k,l} h[k,l]\, I[m+k, n+l]$$

Y = filter2(h, I); or Y = imfilter(I,h);

2D Convolution (Filtreyi uygulayarak görüntüde değişimler yapmak.)

$$Y[m,n] = \sum_{k,l} h[k,l]\, I[m-k, n-l]$$

Y = conv2(h, I); or h = imfilter(I, h, 'conv');

conv2(I,h) = filter2(rot90(h,2), I)

imfilter

- Lineer filters can be implemented in MATLAB by using imfilter function.

Y = imfilter(I, h, mode, boundary_options, size_options);

- I → input image
- h → filter mask
- mode → conv / korelasyon → değişte konvolüsyon yapılacak
- boundary_options → (bulanıklaştırma işin kenar pikseller kaybedilebiliyor. Bunun için 'symmetric' (simetriğini ekler), 'replicate' (aynısını koyar), 'circular' yapılabilir

size options=
'full' → orjinal görüntün pad edilmiş / extend edilmiş boyutta olur.
'same' → output görüntü, input görüntü boyutunda olur.

fspecial

- fspecial: simply create predefined 2D filters
  h = fspecial (type, parameters)

type şunlar alabilir;
- 'average' → Averaging filter
- 'disk' → circular averaging filter
- 'gaussian' → gaussian low-pass filter
- 'laplacian' → 2D laplacian operator
- 'log' → laplacian of Gaussian (LoG) filter
- 'motion' → Approximates the linear motion of camera
- 'prewitt' and 'sobel': horizontal edge-emphasizing filters
- 'unsharp': unsharp contrast enhancement filter.

Örnekler

Image Filtering.m adında script açılır.

```
%% Read Image
im = imread('panda.jpg');
figure; imshow(im);
```

// genelde greyscale üzerinden filtreleme yapılır. Öbür türlü R,G,B için
// ayrı ayrı filtreme yapılmalı.

```
im = rgb2gray(im);
im = im2double(im);
```

```
%% low-pass filters

h_box = fspecial('average', [13 13]); // genelde tek sayı verilir ki
                                      // ortadaki sayı alınabilsin.

// kendimiz üretmek istersek

%h_box = ones(13)./(13*13);

im_box = imfilter(im, h_box, 'conv');
figure; imshow(im_box);
                                      (sigma arttıkça bulanıklık da artar.)
h_gauss = fspecial('gaussian', [13 13], 0.5);
im_gaus = imfilter(im, h_gauss, 'conv');
figure; imshow(im_gauss);


%% high-pass filters

h_lap = [0 -1 0; -1 4 -1; 0 -1 0]; // Laplacian filter.
im_lap = imfilter(im, h_lap, 'conv');
figure; imshow(im_lap);          -1 8 daha iyi ama gürültü oluyor.
```

* Command window'a

```
>> sum(h_gaus(:))  → 1 çıkar
         h_box     → 1
         h_lap     → 0 çıkar
```

laplacian kullanılarak görüntüyü keskinleştirme:

```
im_shap = im + im_lap;
figure; imshow(im_sharp);
```

%% Edge Detection

```
h-sobel = fspecial ('sobel');                      → transpose
im-sobelH = imfilter (im, h-sobel, 'conv');
im-sobelV = imfilter (im, h-sobel', 'conv');

figure; imshow (im-sobelH);
figure; imshow (im-sobelV);

im-edge = im-sobelH + im-sobelV;
figure; imshow (im-edge);  // hem yatay hem dikey kenar tespiti.
```

image gradient magnitude = $\sqrt{g_y^2 + g_x^2}$ yönü = $\theta = \tan^{-1}\left[\frac{g_y}{g_x}\right]$

```
 duman gibi neon    → yön anlatılır
[magnitude, orientation] = imgradient (im-sobelH, im-sobelV);
  figure; imshow (magnitude);
  figure; imshow (orientation, []);  → ölçekleme yapar
```

%% Other Filters

(bu hareketin büyüklüğünü belirtir.)

length ↓, angle (saat yönünün tersi) ↓

```
h-motion = fspecial ('motion', 20, 45);
im-motion = imfilter (im, h-motion, 'conv');
figure; imshow (im-motion);
```

90 için yukarıdan aşağıya hareket olur.

0 derece sağdan sola

Bölüm 7 Binary Images

- In a binary image, each pixel assumes one of only two discrete values: 1 or 0.

- A binary image is stored as a logical array in MATLAB

Thresholding

- Binary images are often produced by thresholding a grayscale or color image, in order to seperate an object in the image from the background.

- Thresholding provides an easy and convenient way to seperate out the regions of the image corresponding to objects in which we are interested, from the regions of the image that correspond to background

- In simple implementation, each pixel in the image is compared with a threshold. If the pixel's intensity is higher than the threshold, the pixel is set to white in the output. If it less than the threshold, it is set to black

Threshold tespiti için çeşitli yöntemler mevcut.
Bunlardan biri Otsu's methodu.

contribute= katkıda bulunur.

Automatic Thresholding: Otsu's Method

labbookpages 'den okumayı önerir

- Assumption: the histogram is bimodal
- Method: find the threshold t that minimizes the weighted sum of within-group variances for two groups that result from separating the grey tones at value t.

```
>> t= graythresh(im) % image threshold using Otsu's method
```

Mathematical Morphology

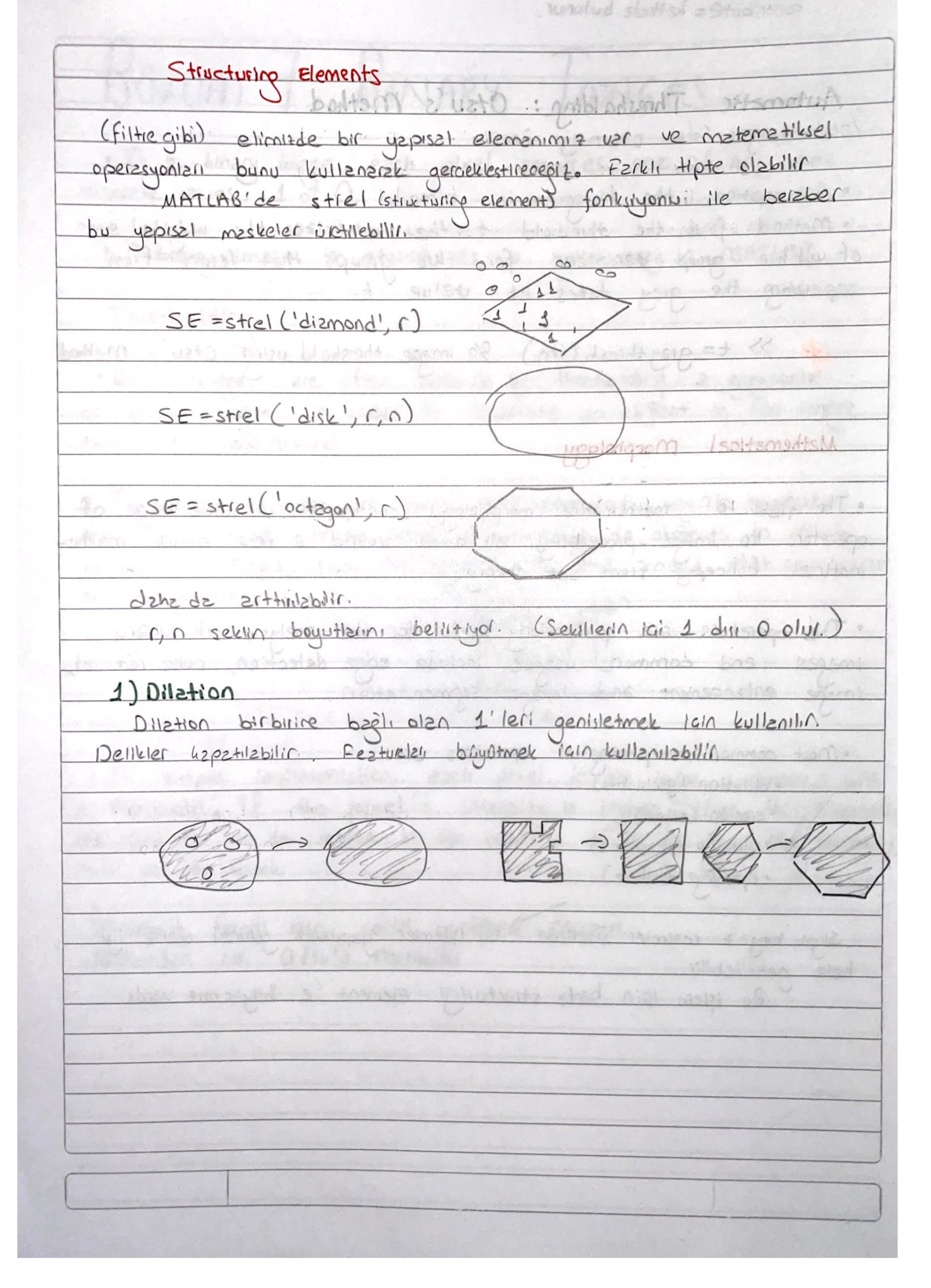
- The field of mathematical morphology contributes a wide range of operators to image processing, all based around a few simple mathematical concepts from set theory.

- The operators are particularly useful for the analysis of binary images and common usages include edge detection, noise removal, image enhancement and image segmentation.

- Most common morphological operators are:
  - dilation (genişletme)
  - erosion (silme)
  - closing (kapama)
  - opening (silme)

Siyah beyaz resimler üzerinde bu işlemler yapılarak görüntü daha iyi hale getirilebilir.

Bu işlem için basta structuring element 'e ihtiyacımız vardır.



Structuring Elements

(filtre gibi) elimizde bir yapısal elemanımız var ve matematiksel operasyonları bunu kullanarak gerçekleştireceğiz. Farklı tipte olabilir
MATLAB'de strel (structuring element) fonksiyonu ile beraber bu yapısal maskeler üretilebilir.

SE = strel('diamond', r)

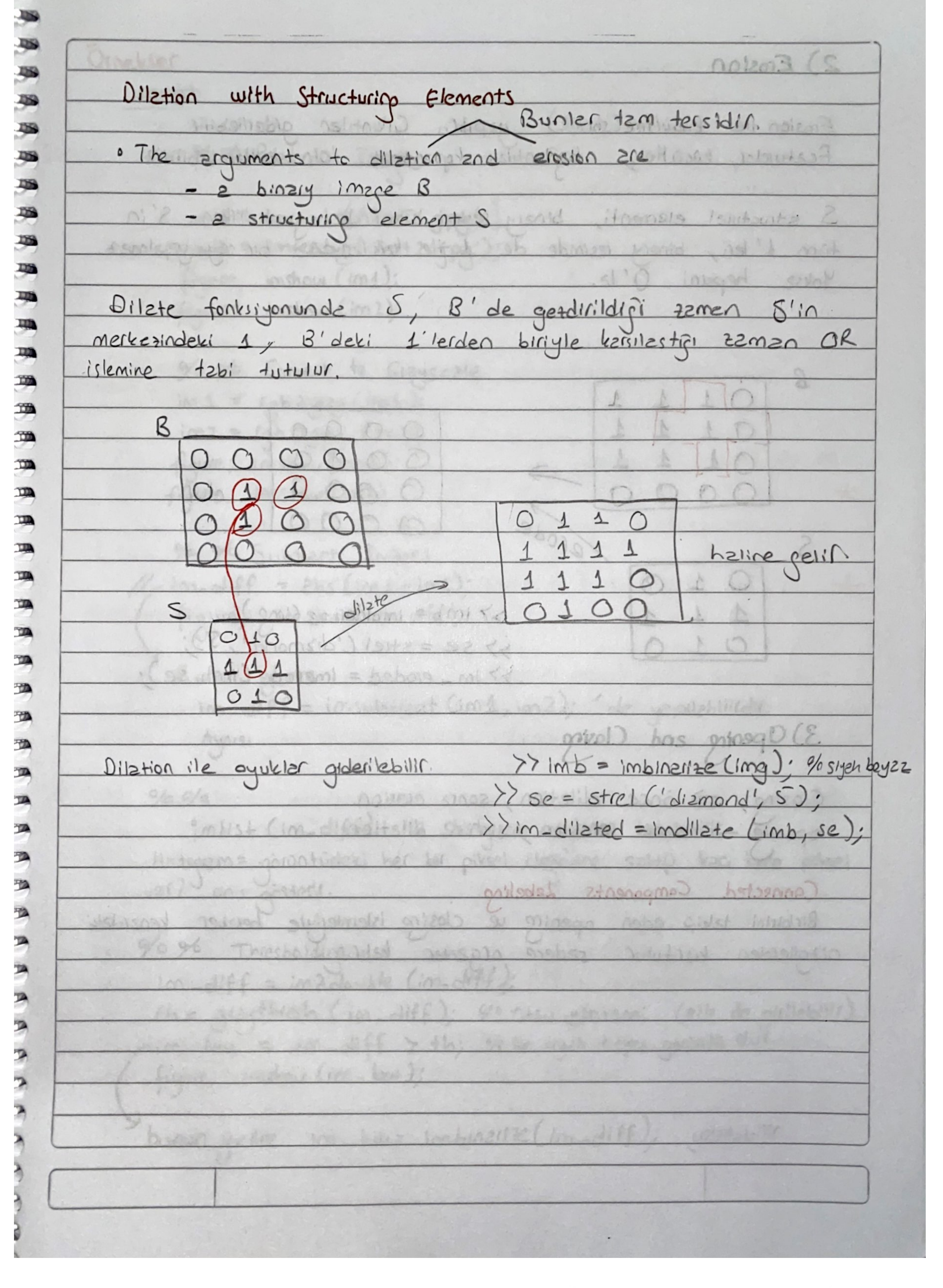
SE = strel('disk', r, n)

SE = strel('octagon', r)

daha da arttırılabilir.
r, n şeklin boyutlarını belirtiyor. (Şekillerin içi 1 dışı 0 olur.)

1) Dilation

Dilation birbirine bağlı olan 1'leri genişletmek için kullanılır. Delikler kapatılabilir. Featureları büyütmek için kullanılabilir.



Dilation with Structuring Elements

- The arguments to dilation and erosion are (Bunlar tam tersidir.)
  - a binary image B
  - a structuring element S

Dilate fonksiyonunda S, B'de gezdirildiği zaman S'in merkezindeki 1, B'deki 1'lerden biriyle karşılaştığı zaman OR işlemine tabi tutulur.

B

0	0	0	0
0	1	1	0
0	1	0	0
0	0	0	0

S

0	1	0
1	1	1
0	1	0

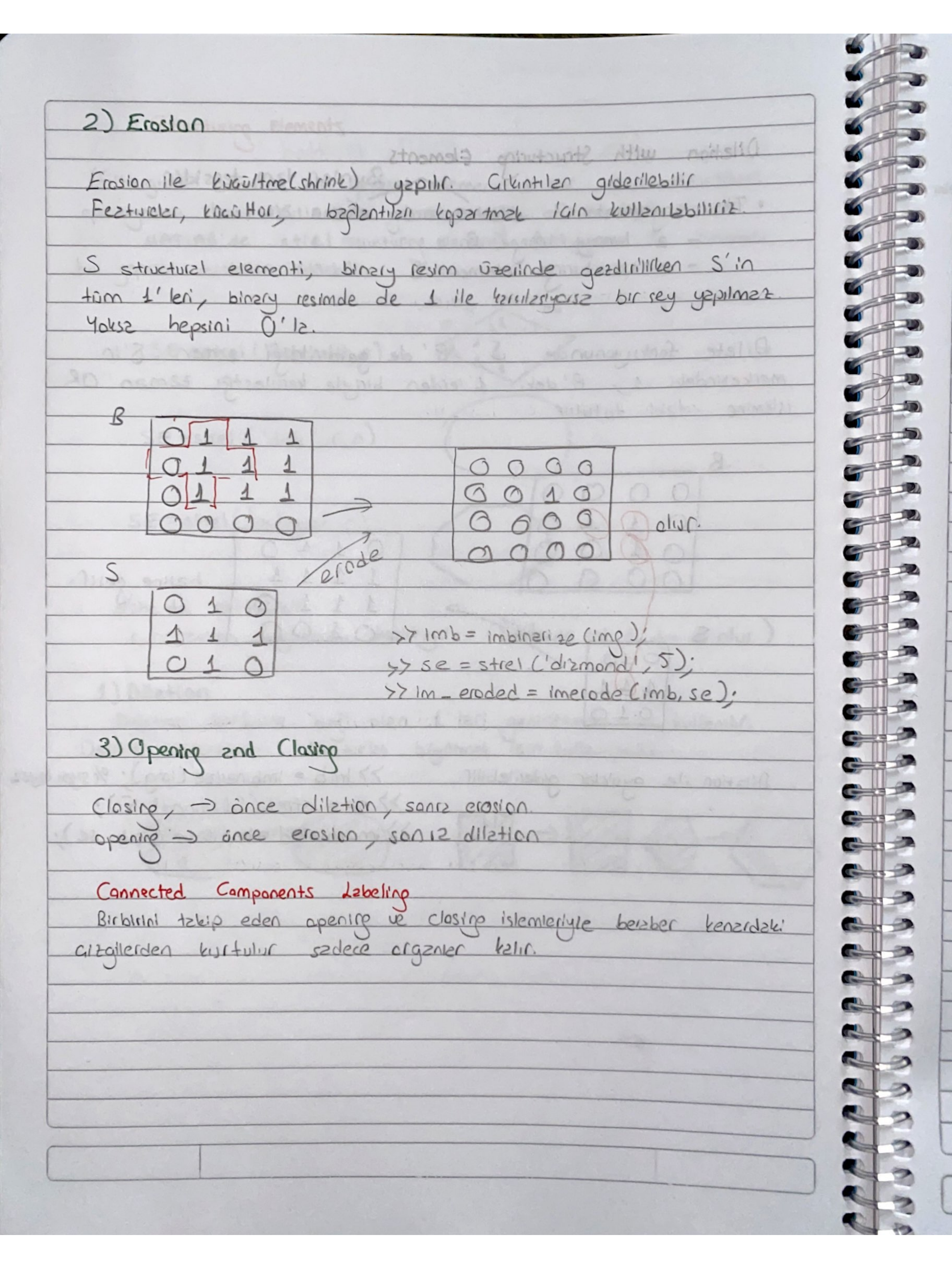
dilate →

0	1	1	0
1	1	1	1
1	1	1	0
0	1	0	0

haline gelir.

Dilation ile oyuklar giderilebilir.

```
>> imb = imbinarize(img); % siyah beyaz
>> se = strel('diamond', 5);
>> im_dilated = imdilate(imb, se);
```



2) Erosion

Erosion ile küçültme (shrink) yapılır. Çıkıntılar giderilebilir. Featurelar küçültülür, bağlantıları koparmak için kullanılabiliriz.

S structural elementi, binary resim üzerinde gezdirilirken S'in tüm 1'leri, binary resimde de 1 ile karşılaşıyorsa bir şey yapılmaz. Yoksa hepsini 0'la.

B

0	1	1	1
0	1	1	1
0	1	1	1
0	0	0	0

S

0	1	0
1	1	1
0	1	0

erode →

0	0	0	0
0	0	1	0
0	0	0	0
0	0	0	0

olur.

```
>> imb = imbinarize(img);
>> se = strel('diamond', 5);
>> im_eroded = imerode(imb, se);
```

3) Opening and Closing

Closing → önce dilation, sonra erosion.
opening → önce erosion, sonra dilation

Connected Components Labeling

Birbirini takip eden opening ve closing işlemleriyle beraber kenardaki çizgilerden kurtulur sadece organlar kalır.

Örnekler

script

```
%% Read in Images

im1 = imread ('Test1.jpg');
im2 = imread ('Test2.jpg');
figure; imshow (im1);
figure; imshow (im2);

%% Convert to Grayscale
im1 = rgb2gray (im1);
im2 = rgb2gray (im2);
figure; imshow (im1);
figure; imshow (im2);

%% Subtract Images
im_diff = abs (im1 - im2);
figure; imshow (im_diff);
```

Bunun yerine
im_diff = imsubtract (im1, im2); 'de yapılabilirdir.
Aynısı.

%%
imhist (im_diff) ile histogram değerine bakılabilir.
Histogram = görüntüdeki her bir piksel değerine sahip kaç tane piksel var? onu gösterir.

```
%% Thresholding
im_diff = im2double (im_diff);
th = graythresh (im_diff); % Otsu yöntemi (elle de girilebilir)
im_bw = im_diff > th; % ile siyah beyaz görüntü olur.
figure; imshow (im_bw);
```

bunun yerine im_bw = imbinarize (im_diff); yazılabilir.

```
%% Fill in Regions

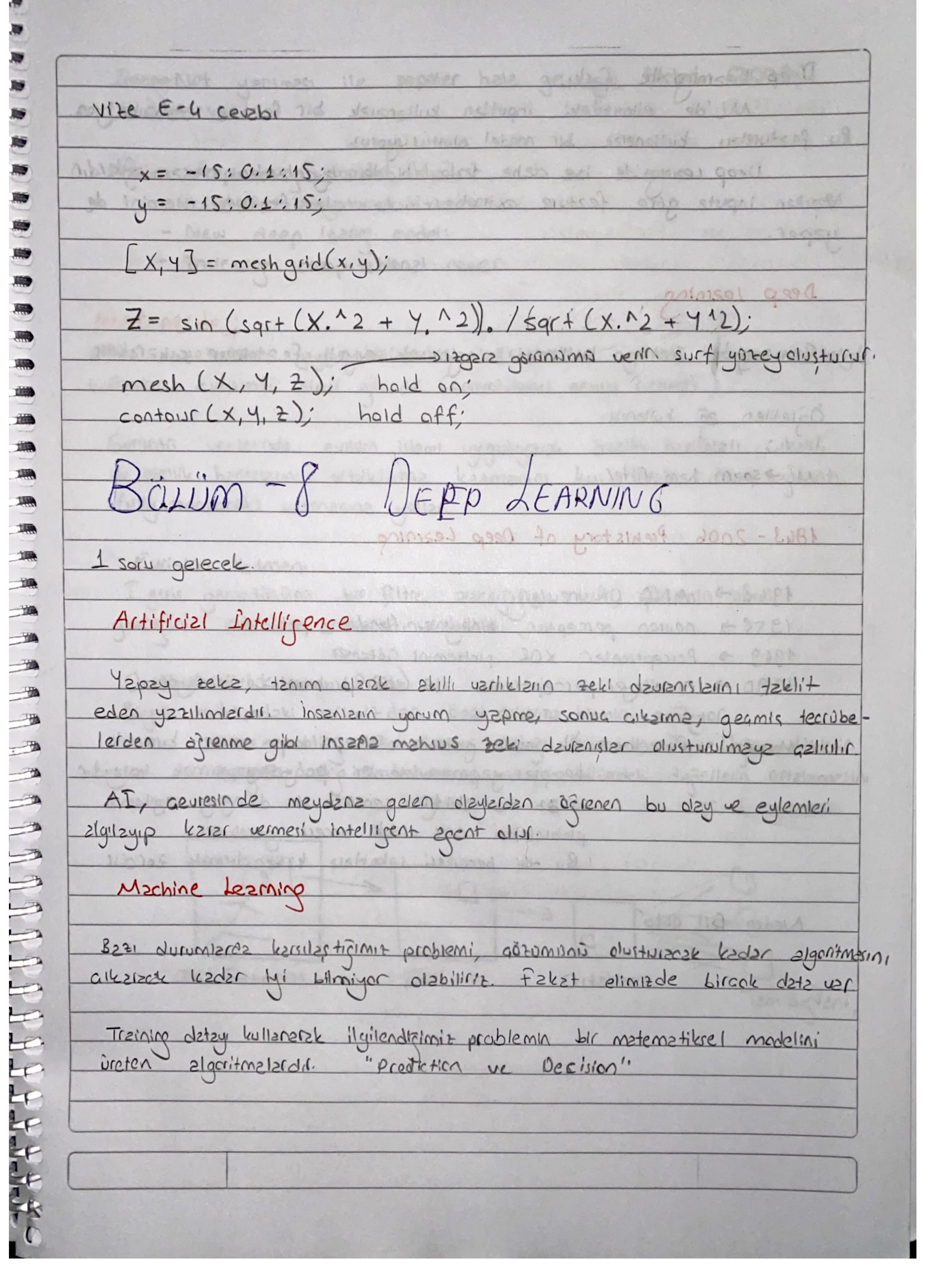
se = strel('disk', 3); % Çapı 3  7 ye 7 'liğe karşılık gelir.
im_open = imopen(im_bw, se); % daraltma yapar. imclose yapılabilir
                                                  açma için
figure; imshow(im_open);

% Gürültülerden kurtulmak için; morfolojik operatör kullanılmış olur.
                   'disk', 1
                   imopen yapılır.
                                                   pixel
im_area = bwareaopen(im_bw, 15); // 15 'den daha küçük
                                         olan bölgeler gider.

%% Region Properties
*
    im_stats = regionprops(im_open, 'MajorAxisLength');
*
    im_length = [im_stats.MajorAxisLength];
                          80 üzeri dersem bu bir nesnedir denilebilir.
                          (4 bölge var.)
    idx = im_length > 80;
    im_stats_final = im_stats(idx);
    disp(im_stats_final);

%% Determine if change is significant

    if isempty(im_stats_final)
        disp('Nothing here');
    else
        disp('Something here');
    end
```



Vize E-4 cevabı

```
x = -15:0.1:15;
y = -15:0.1:15;

[X,Y] = meshgrid(x,y);

Z = sin(sqrt(X.^2 + Y.^2))./sqrt(X.^2 + Y^2);

mesh(X,Y,Z);      hold on;
contour(X,Y,Z);   hold off;
```

mesh → ızgara görünümü verir. surf yüzey oluşturur.

BÖLÜM - 8 DEEP LEARNING

1 soru gelecek.

Artificial Intelligence

Yapay zeka, tanım olarak akıllı varlıkların zeki davranışlarını taklit eden yazılımlardır. İnsanların yorum yapma, sonuç çıkarma, geçmiş tecrübelerden öğrenme gibi insana mahsus zeki davranışlar oluşturulmaya çalışılır.

AI, çevresinde meydana gelen olaylardan öğrenen bu olay ve eylemleri algılayıp karar vermesi intelligent agent olur.

Machine Learning

Bazı durumlarda karşılaştığımız problemi, çözümünü oluşturacak kadar algoritmasını çıkaracak kadar iyi bilmiyor olabiliriz. Fakat elimizde birçok data var.

Training datayı kullanarak ilgilendiğimiz problemin bir matematiksel modelini üreten algoritmalardır. "Prediction ve Decision"

Deep Learning ile farkı;

ML'de elimizdeki inputları kullanarak bir feature set çıkarıyoz. Bu featureları kullanarak bir model oluşturuyoruz.

Deep Learning'de ise daha fazla hidden layerlardan oluşan ağlardır. Verilen inputa göre feature extraction'ı da classification işlemini de yapar.

Deep Learning

Çok sayıda katman kullanarak yüksek seviyeli featurelar çıkarabilir.

Öğretilen ağ kullanılır.

ML → spam, ham filter.

1943 - 2006 Prehistory of Deep Learning

1943 → AND OR neuron
1958 → neuron perceptron, binary sınıflandırma problemi.
1969 → Perceptronlar XOR problemini çözemez.
1990 → multilayer perceptronlar ile XOR problemi çözülebiliyor.

Moravec's Paradox = İlk akla gelenin aksine, karar verme karmaşık hesaplar yapma işlemleri pc'ye yaptırmak kolaydır ve az maliyetlidir. sensörlü dış dünyayı algılama problemleri çok ciddi beceri gerektirir. Bu tür becerileri robotlara kazandırmak zordur.

Neden fail oldu?

ImageNet yarışması ile popüler hale gelmeye başladı. (2009)
Alexnet var.

Neden deeplearning popüler oldu?
- Verisetlerinin boyutunun artması
- New deep learning models
- Increasing computational power

MATLAB'de;
DNN modellerinden yaygın olarak kullanılan özellikle görüntü işleme modelleriyle kullanılan evrişimli sinir ağlarıdır. (convolutional neural network.)

Görüntü verilerinde evrişim işlemi uygulayarak özellik haritaları çıkarır. Bu özellik haritalarını ortaklama katmanını kullanarak tam bağlantı (fully connected) katmanına yollar.

Evrişim Katmanı
I giriş görüntüsünü bir filtre aracılığıyla tarar, filtrenin nasıl geçeceğini parametre olarak verebiliriz.

Ortaklama Katmanı (POOL)
Büyük miktardaki verimizden özet çıkarma işlemini sağlıyor.
max-pooling → bulunmuş olduğu kısımdaki max değeri döndürür.
average-pooling → bulunmuş olduğu bölümdeki değerlerin ortalamasını buluyor.

Giriş Görüntüsü — Evrişim — Ortaklama — Tam Bağlantı

Tam Bağlantı

Yapay sinir ağ modelidir. Sonuç çıkarmak için kullanılır.

Aktivasyon Fonksiyonları

Elde edilen verilerin Doğrusal olmasını engellemek amacıyla aktivasyon fonksiyonları kullanılır. ReLU Sızıntı Relu ELU

Softmax → Sınıflandırma tarzı uygulamalarda kullanılır.

ÖR

script açılır.

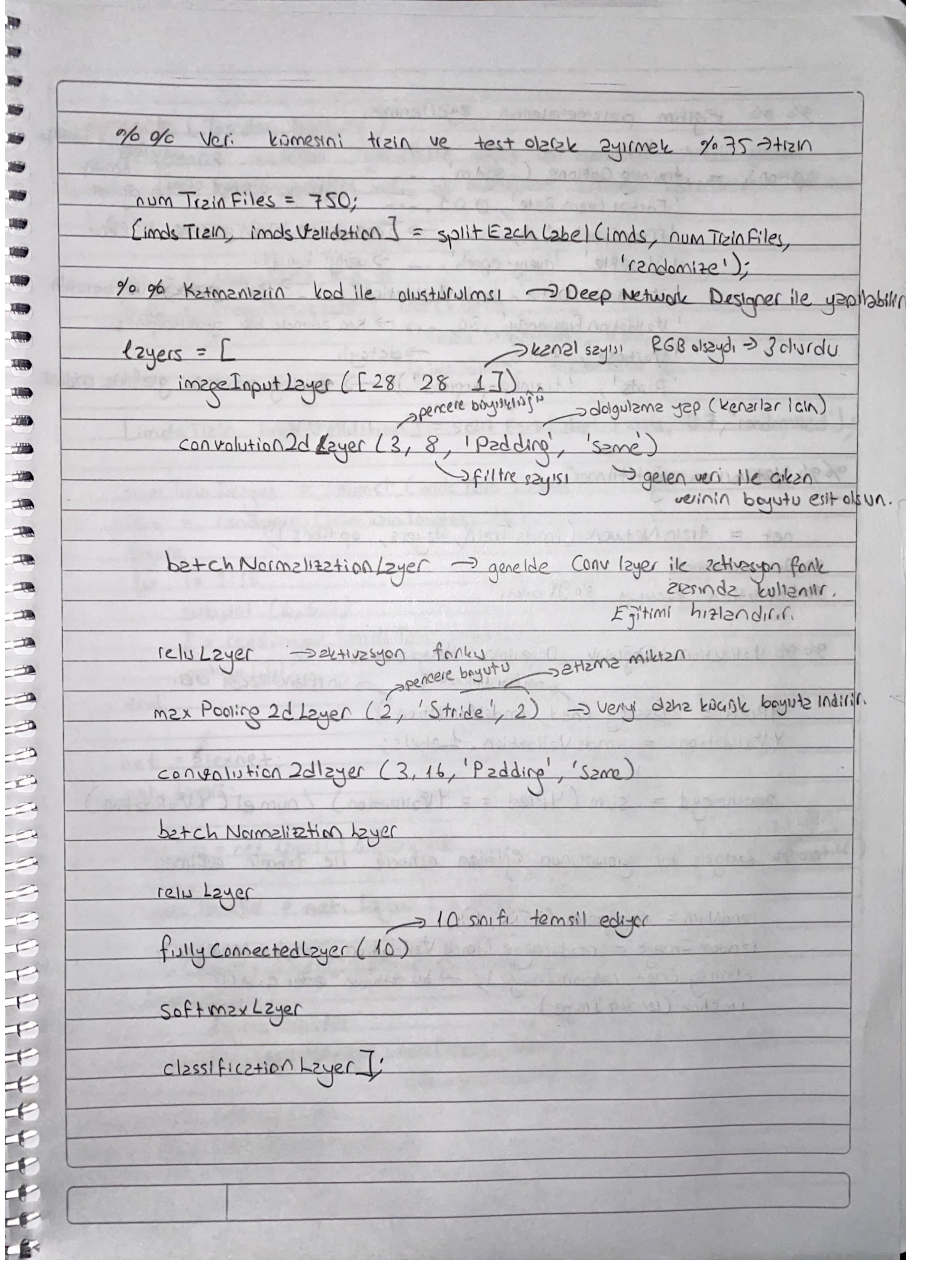
```
%% datasetin yüklenmesi

digitDatasetPath = fullfile(matlabroot, 'toolbox', 'nnet', 'nndemos'
    'nndatasets', 'DigitDataset');
imds = imageDatastore(digitDatasetPath,
    'IncludeSubfolders', true, 'LabelSource', 'foldernames');

%% datasetten örnek görüntülerin gösterilmesi
figure;
perm = randperm(10000, 20);
for i = 1:20
    subplot(4, 5, i);
    imshow(imds.Files{perm(i)});
end

%% datasetle alakalı bilgiler; (hangi rakamdan kaçar tane)
img = readImage(imds, 1);
size(img)
labelCount = countEachLabel(imds)
```

(Kenar notu, kodun ilk bloğunun yanında: Bunu yazmaz)



%% Veri kümesini train ve test olarak ayırmak %75 → train

```
numTrainFiles = 750;
[imdsTrain, imdsValidation] = splitEachLabel(imds, numTrainFiles,
                                   'randomize');
```

%% Katmanların kod ile oluşturulması → Deep Network Designer ile yapılabilir

```
layers = [
    imageInputLayer([28 28 1])          1 → kanal sayısı   RGB olsaydı → 3 olurdu

    convolution2dLayer(3, 8, 'Padding', 'same')
                       3 → pencere boyutu
                       8 → filtre sayısı
                       'Padding' → dolgulama yap (kenarlar için)
                       'same' → gelen veri ile çıkan verinin boyutu eşit olsun.

    batchNormalizationLayer   → genelde Conv layer ile aktivasyon fonk
                                arasında kullanılır.
                                Eğitimi hızlandırır.

    reluLayer                 → aktivasyon fonku

    maxPooling2dLayer(2, 'Stride', 2)   → veriyi daha küçük boyuta indirir.
                       2 → pencere boyutu
                       'Stride', 2 → atlama miktarı

    convolution2dLayer(3, 16, 'Padding', 'same')

    batchNormalizationLayer

    reluLayer

    fullyConnectedLayer(10)   10 → 10 sınıfı temsil ediyor

    softmaxLayer

    classificationLayer];
```

%% Eğitim parametrelerinin ayarlanması

```
options = trainingOptions('sgdm', ...
    'InitialLearnRate', 0.01, ... → Başlangıç öğrenme oranı (Büyük seçilirse hemen öğrenir.)
    'MaxEpochs', 4, ... → Tüm verinin elden geçirilme 4 kere
    'Shuffle', 'every-epoch', ... → veriyi karıştırır
    'ValidationData', imdsValidation, ... → test yapılacak veri belirtilir
    'ValidationFrequency', 30, ... → Kaç adımda bir göstereceğimiz
    'Verbose', false, ... → detaylı
    'plots', 'training-progress'); → training aşamasını grafikle çizdirir.
```

%% Networkun eğitilmesi

```
net = trainNetwork(imdsTrain, layers, options);
% Bunu çalıştırınca
% Validation accuracy %98 çıktı.
```

%% Validation Doğruluk Değerinin Hesaplanması

```
                 sınıflandırılacak ağ   sınıflandırılacak veri
YPred = classify(net, imdsValidation);
YValidation = imdsValidation.Labels;

accuracy1 = sum(YPred == YValidation) / numel(YValidation)
```

%% Rastgele bir görüntünün eğitilen network ile tahmin edilmesi

```
randNum = randperm(2500,1);
randomImage = readimage(imdsValidation, randomNumber);
classify(net, randomImage); → bu tahmin eder çıktılar.
imshow(randomImage)
```

Alexnet (Transfer learning)

Derin öğrenme modelleri genellikle büyük veri gerektirir. Ancak yeterli veri yoksa hazır modeller var. Bu modellerin eğitilmiş katmanlarını kullanabiliriz.

```
unzip ('MerchData.zip');
imds = imageDatastore ('MerchData', ...
                       'Include Subfolders', true, ...
                       'LabelSource', 'foldernames');

[imdsTrain, imdsValidation] = splitEachLabel (imds, 0.7, 'randomized');

numTrainImages = numel (imdsTrain.Labels);
idx = randperm(numTrainImages, 16);
figure;
for i = 1:16
    subplot (4,4,i)
    I = readimage (imdsTrain, idx(i));
    imshow (I);
end

net = alexnet;
net.Layers.

inputSize = net.Layers(1).InputSize

layersTransfer = net.Layers (1:end-3);
```

→ Son 3 kısım atılır. (23'den sonrası atılır.)

```
numClasses = numel( categories (imdsTrain.Labels))
layers = [
    layersTransfer
    fullyConnectedLayer (numClasses, 'WeightLearnRateFactor', 20,
                         'BiasLearnRateFactor', 20)
    softmaxLayer
    classificationLayer ];
```

% image augmenter, → elimizdeki veriyi arttırma işlemidir.

```
pixelRange = [-30 30];
imageAugmenter = imageDataAugmenter( ...
                'RandXReflection', true, ...
                'RandXTranslation', pixelRange, ...
                'RandYTranslation', pixelRange);

augimdsTrain = augmentedImageDatastore(inputSize(1:2), imdsTrain, ...
               'DataAugmentation', imageAugmenter);

augimdsValidation = augmentedImageDatastore(inputSize(1:2),
                                            imdsValidation);

options = trainingOptions('sgdm', ...
          'MiniBatchSize', 10, ...
          'MaxEpochs', 6, ...
          'InitialLearnRate', 1e-4, ...
          'ValidationData', augimdsValidation, ...
          'ValidationFrequency', 3, ...
          'ValidationPatience', Inf, ...
          'Verbose', false, ...
          'Plots', 'training-progress');

netTransfer = trainNetwork(augimdsTrain, layers, options);

[YPred, scores] = classify(netTransfer, augimdsValidation);
```

```
idx = randperm(numel(imdsValidation.Files),4);
figure;
for i= 1:4
    subplot(2,2,i);
    I = readImage(imdsValidation, idx(i));
    imshow(I);
    label = YPred(idx(i));
    title(string(label));
end

YValidation = imdsValidation.Labels;
accuracy = mean(YPred == YValidation)
```